

μ PD75402A
4-BIT SINGLE-CHIP MICROCOMPUTER

μ PD75402A
 μ PD75P402

MS-DOS is a trademark of Microsoft Corporation.
PC/AT, PC DOS are trademarks of IBM Corporation.

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

The devices listed in this document are not suitable for use in the field where very high reliability is required including, but not limited to, aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. If customers intend to use NEC devices for above applications or those intend to use "Standard" quality grade NEC devices for the application not intended by NEC, please contact our sales people in advance.

Application examples recommended by NEC Corporation

Standard : Computer, Office equipment, Communication equipment, Test and Measurement equipment, Machine tools, Industrial robots, Audio and Visual equipment, Other consumer products, etc.

Special : Automotive and Transportation equipment, Traffic control systems, Antidisaster systems, Anti-crime systems, etc.

Major Revisions in This Version

Section	Description
P.117	Amendment: Fig. 5-52 "Data Transmission from Slave Device to Master Device"
P.179 to 181	Change: Appendix B "Development Tools"

The mark ★ shows main revised points.

PREFACE

USER

This manual is intended for user engineers who wish to understand the μ PD75402A's, 75P402's functions and design an application system using them.

OBJECTIVE

The objective of this manual is for the user to understand the μ PD75402A's, 75P402's hardware functions shown below.

COMPOSITION

This manual is composed roughly of the following contents.

- General description
- Pin functions
- Internal block functions
- Interrupt
- Other internal peripheral functions
- Instruction functions

READING

The reader of this manual must have general knowledge of electricity, logic circuitry and microcomputers.

- Use as the μ PD75P402 manual
→This manual describes the μ PD75402A as the representative model unless there are particular differences in functions. If using it as the μ PD75P402 manual, the μ PD75402A should be read as the μ PD75P402.
- Checking an instruction function with a known mnemonic
→**Appendix D. "Instructions Index (in Alphabetical Order)"** should be used.
- Checking an instruction with an unknown mnemonic but with a roughly known function
→Check the instruction's mnemonic in **9.2 "Instruction Set and Its Operation"** and then check the function in **9.4 "Instruction Functions and Application"**.
- Roughly understanding the μ PD75402A's, 75P402's functions
→Follow the contents.

LEGEND

Data notation weight	:	High-order digit on the left, low-order digit on the right
Notation of active low	:	$\overline{\text{xxx}}$ (line over pin, signal name)
Memory map address	:	Upper__low order, lower__high order
*	:	Explanation of * in text
Note	:	Content to be read carefully
Remarks	:	Supplementary explanation of text
Numeric notation	:	Binary ... xxxx or xxxxB Decimal ... xxxx Hexadecimal ... xxxxH

Related Documentation

Device Related Documents

Document Name	Document Number
User's Manual	IEU-644
Instruction Application Table	IEM-5504
Application Note	IEA-638
75X Series Selection Guide	IF-151

Development Tool Related Documents

Document Name		Document Number	
Hardware	IE-75000-R/IE-75001-R User's Manual	EEU-846	
	IE-75000-R-EM User's Manual	EEU-673	
	EP-75402C-R User's Manual	EEU-701	
	EP-75402GB-R User's Manual	EEU-702	
	PG-1500 User's Manual	EEU-651	
Software	RA75X Assembler Package I	Operation	EEU-731
	User's Manua	Language	EEU-730
	PG-1500 Controller User's Manual	EEU-704	

Other Related Documents

Document Name	Document Number
Package Manual	IEI -635
Surface Mount Technology Manual	IEI -616
Quality Grade on NEC Semiconductor Devices	IEI -620
NEC Semiconductor Device Reliability & Quality Control	IEM -5068
Electrostatic Discharge (ESD) Test	MEM -539
Semiconductor Devices Quality Control Guarantee Guide	MEI -603
Microcomputer Related Products Guide. Other Manufacturers	MEI -604

Note The above documents are subject to change without notice.
The latest documents should be used for design purposes, etc.

CONTENTS

CHAPTER 1. GENERAL	1
1.1 OUTLINE OF FUNCTIONS	2
1.2 ORDERING INFORMATION AND QUALITY GRADE	3
1.3 DIFFERENCES BETWEEN μPD75402A AND μPD75402, 75P402	4
1.4 BLOCK DIAGRAM	5
1.5 PIN CONFIGURATION	6
1.5.1 28-Pin Plastic DIP (600 mil), Shrink DIP (400 mil)	6
1.5.2 44-Pin Plastic QFP (\square 10 mm)	8
CHAPTER 2. PIN FUNCTIONS	10
2.1 μPD75402A PIN FUNCTION LIST	11
2.1.1 Port Pin List	11
2.1.2 List of Pins Other than Port Pins	12
2.2 NORMAL OPERATING MODE	13
2.2.1 P00 to P03 (Port 0), P10, P12 (Port 1)	13
2.2.2 P20 to P23 (Port 2), P30 to P33 (Port 3), P50 to P53 (Port 5), P60 to P63 (Port 6)	14
2.2.3 \overline{SCK} , SO/SB0, SI	14
2.2.4 INT0	14
2.2.5 INT2	14
2.2.6 PCL	14
2.2.7 X1, X2	15
2.2.8 \overline{RESET} (Reset)	15
2.2.9 V_{DD}	15
2.2.10 V_{SS}	15
2.3 PROM MODE	16
2.3.1 A0 to A14 (Address)	16
2.3.2 O0 to O7 (Data)	16
2.3.3 \overline{CE} (Chip Enable)	16
2.3.4 \overline{OE} (Output Enable)	16
2.3.5 V_{PP}	16
2.3.6 V_{DD}	16
2.3.7 V_{SS}	16
2.4 PIN INPUT/OUTPUT CIRCUITS	17
2.5 UNUSED PIN TREATMENT	20
2.6 NOTES ON USE OF P00 PIN AND \overline{RESET} PIN	20
CHAPTER 3. FEATURES OF ARCHITECTURE AND MEMORY MAP	21
3.1 DATA MEMORY BANK CONFIGURATION AND ADDRESSING MODES	21
3.1.1 Data Memory Bank Configuration	21
3.1.2 Data Memory Addressing Modes	24
3.2 MEMORY-MAPPED I/O	28

CHAPTER 4. INTERNAL CPU FUNCTIONS	31
4.1 PROGRAM COUNTER (PC)	31
4.2 PROGRAM MEMORY (ROM)	32
4.3 DATA MEMORY (RAM)	33
4.4 GENERAL REGISTER	35
4.5 ACCUMULATOR	36
4.6 STACK POINTER (SP)	37
4.7 PROGRAM STATUS WORD (PSW)	39
CHAPTER 5. PERIPHERAL HARDWARE FUNCTIONS	41
5.1 DIGITAL INPUT/OUTPUT PORTS	41
5.1.1 Digital Input/Output Port Types, Characteristics and Configuration	42
5.1.2 Input/Output Mode Setting	46
5.1.3 Digital Input/Output Port Handling Instructions	47
5.1.4 Digital Input/Output Port Operations	49
5.1.5 Internal Pull-Up Resistors	51
5.1.6 Digital Input/Output Port Input/Output Timing	53
5.2 CLOCK GENERATION CIRCUIT	54
5.2.1 Clock Generation Circuit Configuration	54
5.2.2 Clock Generation Circuit Function and Operation	55
5.2.3 CPU Clock Setting	59
5.2.4 Differences Between μ PD75402A and μ PD75402	61
5.3 CLOCK OUTPUT CIRCUIT	63
5.3.1 Clock Output Circuit Configuration	63
5.3.2 Clock Output Mode Register (CLOM)	64
5.3.3 Clock Output Procedure	65
5.3.4 Example of Remote Control Application	65
5.4 BASIC INTERVAL TIMER	66
5.4.1 Basic Interval Timer Configuration	66
5.4.2 Basic Interval Timer Mode register (BTM)	67
5.4.3 Basic Interval Timer Operation	68
5.4.4 Examples of Basic Interval Timer Applications	69
5.5 SERIAL INTERFACE	70
5.5.1 Serial Interface Functions	70
5.5.2 Serial Interface Configuration	71
5.5.3 Register Functions	74
5.5.4 Operation-Halt Mode	83
5.5.5 3-Wire Serial I/O Mode Operation	84
5.5.6 SBI Mode Operation	93
CHAPTER 6. INTERRUPT FUNCTIONS	126
6.1 INTERRUPT CONTROL CIRCUIT CONFIGURATION	126
6.2 INTERRUPT SOURCE TYPES AND VECTOR TABLE	128
6.3 INTERRUPT CONTROL CIRCUIT HARDWARE	129
6.4 INTERRUPT SEQUENCE	134

6.5	MACHINE CYCLES BEFORE INTERRUPT SERVICING	135
6.6	INTERRUPT APPLICATIONS	137
CHAPTER 7. STANDBY FUNCTION		141
7.1	STANDBY MODE SETTING AND OPERATION STATES	142
7.2	STANDBY MODE RESET	143
7.3	OPERATION AFTER STANDBY MODE RESET	145
7.4	STANDBY MODE APPLICATION	145
CHAPTER 8. RESET FUNCTION		146
CHAPTER 9. INSTRUCTION SET		148
9.1	SPECIAL INSTRUCTION	149
9.1.1	Bit Operation Instructions	149
9.1.2	Stack Instructions	149
9.1.3	Base Correction Instructions	150
9.1.4	Skip Instruction and Number of Machine Cycles Required by Skip	150
9.2	INSTRUCTION SET AND ITS OPERATION	151
9.3	OPERATION CODE OF EACH INSTRUCTION	156
9.4	INSTRUCTION FUNCTIONS AND APPLICATION	159
9.4.1	Move Instructions	159
9.4.2	Table Reference Instructions	162
9.4.3	Arithmetic and Logic Instructions	163
9.4.4	Accumulator Operation Instructions	165
9.4.5	Increment/Decrement Instructions	166
9.4.6	Compare Instructions	167
9.4.7	Carry Flag Operation Instructions	168
9.4.8	Bit Manipulation Instructions	169
9.4.9	Branch Instructions	171
9.4.10	Subroutine Stack Control Instructions	172
9.4.11	Interrupt Control Instructions	174
9.4.12	Input/Output Instructions	175
9.4.13	CPU Control Instructions	176
APPENDIX A. TABLE OF INSTRUCTION USABLE WITH EVAKIT-75X ONLY		177
APPENDIX B. DEVELOPMENT TOOLS		179
APPENDIX C. MASK ROM ORDERING PROCEDURE		182
APPENDIX D. INSTRUCTION INDEX (ALPHABETIC ORDER)		183
APPENDIX E. HARDWARE INDEX (ALPHABETIC ORDER)		184

CONTENTS OF FIGURES

Fig. No	Title	Page
3-1	Static RAM Address Updating Method	25
4-1	Program Counter Configuration	31
4-2	Program Memory Map	32
4-3	Data Memory Map	33
4-4	General Register Configuration	35
4-5	Register Pair Configuration	35
4-6	Accumulators	36
4-7	Stack Pointer Configuration	37
4-8	Data Saved to Stack Memory	38
4-9	Data Restored from Stack Memory	38
4-10	Program Status Word Configuration	39
5-1	Digital Input/Output Port Data Memory Addresses	41
5-2	Configuration of Ports 0 and 1	43
5-3	Configuration of Port 3	44
5-4	Configuration of Ports 2 and 6	45
5-5	Configuration of Port 5	46
5-6	Format of Port Mode Registers	47
5-7	Format of Pull-Up Resistor Specification Register	52
5-8	Pull-Up Resistor Incorporation Switching Timing	52
5-9	Digital Input/Output Port Input/Output Timing	53
5-10	Clock Generation Circuit Block Diagram	54
5-11	Processor Clock Control Register Format	56
5-12	System Clock Oscillation Circuit External Circuitry	57
5-13	Example of Poor Resonator Connection Circuit	57
5-14	Use of Variable Minimum Instruction Execution Time Function	59
5-15	Change of Φ after Power-On Reset	60
5-16	Clock Generation Circuit - Differences between μ PD75402A and μ PD75402	61
5-17	μ PD75402 Processor Clock Control Register Format	62
5-18	Clock Output Circuit Configuration	63
5-19	Clock Output Mode Register Format	64
5-20	Example of Remote Control Application	65
5-21	Basic Interval Timer Configuration	66
5-22	Basic Interval Timer Mode Register Format	67
5-23	Example of SBI System Configuration	71
5-24	Serial Interface Block Diagram	72
5-25	Serial Operating Mode Register (CSIM) Format	75
5-26	Serial Bus Interface Control Register (SBIC) Format	78
5-27	Configuration Around Shift Register	81
5-28	Example of 3-Wire Serial I/O System Configuration	84
5-29	3-Wire Serial I/O Mode Timing	88
5-30	RELT & CMDT Operation	89
5-31	Shift Register (SIO) and Internal Bus Configuration	90

Fig. No.	Title	Page
5-32	Example of SBI Serial Bus System Configuration	93
5-33	SBI Transfer Timing	95
5-34	Bus Release Signal	96
5-35	Command Signal	96
5-36	Address	97
5-37	Slave Selection by Address	97
5-38	Command	98
5-39	Data	98
5-40	Acknowledge Signal	99
5-41	Busy Signal & Ready Signal	100
5-42	RELT, CMDT, RELD & CMDD Operation (Master)	106
5-43	RELT, CMDT, RELD & CMDD Operation (Slave)	106
5-44	ACKT Operation	107
5-45	ACKE Operation	108
5-46	ACKD Operation	109
5-47	BSYE Operation	109
5-48	Pin Configuration Diagram	112
5-49	Address Transmission from Master Device to Slave Device (WUP = 1)	114
5-50	Command Transmission from Master Device to Slave Device	115
5-51	Data Transmission from Master Device to Slave Device.....	116
5-52	Data Transmission from Slave Device to Master Device.....	117
5-53	Example of Serial Bus Configuration	119
5-54	READ Command Transfer Format.....	121
5-55	WRITE & END Command Transfer Format	121
5-56	STOP Command Transfer Format.....	122
5-57	STATUS Command Transfer Format.....	123
5-58	STATUS Command Status Format	123
5-59	RESET Command Transfer Format	124
5-60	CHGMST Command Transfer Format	124
5-61	Master and Slave Operations after an Error.....	125
6-1	Interrupt Control Circuit Block Diagram.....	127
6-2	Interrupt Vector Table	128
6-3	Configuration of INT0 and INT2	130
6-4	INT0 Noise Elimination Circuit Input/Output Timing	131
6-5	INT2 Input Noise Elimination	131
6-6	Edge Detection Mode Register Format	132
6-7	IME Format	132
6-8	Interrupt Servicing Procedure	134
7-1	Standby Mode Reset Operation	144
8-1	Reset Signal Acceptance	146
8-2	Reset at Power-on	146

CONTENTS OF TABLES

Table No.	Title	Page
1-1	Differences Between μ PD75402A and μ PD75402, 75P402	4
2-1	Port Pin List	11
2-2	List of Pins Other than Port Pins	12
2-3	Port 0's, 1's Dual-Function Pins	13
2-4	Pin Input/Output Types	17
3-1	Data Memory Configuration and Address Range in Each Addressing Mode	22
3-2	Addressing Mode List	23
3-3	Applicable Addressing Modes at Peripheral Hardware Operation	28
3-4	μ PD75402A I/O Map	29
4-1	Carry Flag Manipulation Instructions	39
4-2	Interrupt Status Flag Indication Content	40
5-1	Digital Input/Output Port Types and Characteristics	42
5-2	List of Input/Output Pin Handling Instructions	48
5-3	Operations with Input/Output Port Handling Instructions	50
5-4	Internal Pull-up Resister Specification for Each Port	51
5-5	Maximum Time Required for Change of CPU Clock	60
5-6	Serial Clock Selection and Use (in 3-Wire Serial I/O Mode)	89
5-7	Serial Clock Selection and Use (in SBI Mode)	105
5-8	Signals in SBI Mode	110
6-1	Interrupt Request Source Types	128
6-2	Interrupt Request Flag Setting Signal	129
6-3	IST0 Interrupt Servicing Status	133
7-1	Standby Mode Operation States	142
8-1	State of Hardware after Reset	147

CHAPTER 1. GENERAL

The μ PD75402A, 75P402 is a CMOS 4-bit single-chip microcomputer adopting the 75X architecture. With its built-in NEC standard serial bus interface (SBI), it is suitable as a slave microcomputer in a multiprocessor system configuration using the 75X, 78K series as the host microcomputer.

The μ PD75402A has shortened the conventional μ PD75402's minimum instruction execution time to 0.95 μ s. The μ PD75P402 is also capable of high-speed processing.

It is possible to reduce the burden on the host microcomputer by using the μ PD75402A, 75P402 as the slave microcomputer by using the μ PD75402A, 75P402 as the slave microcomputer for decentralized processing. The μ PD75402A, or 75P402 is most suitable for slave processes for the following devices such as the key input control, LED, etc. display control, or remote control send/receive control, etc.

- FAX
- PPC
- Printer
- ECR
- VCR
- Remote control commander

The μ PD75P402 is a product with the μ PD75402A's built-in mask ROM having been replaced with the one-time PROM. It is compatible with the μ PD75402A except for the program memory and the mask option. The pin connection at write is the same as in the standard EPROM μ PD27C256A and also has the same writing characteristics. Therefore, it is possible to write directly using the general-purpose PROM writer. The μ PD75P402 is suitable for preproduction at system development and short-run and multiple-device production.

Name	Program Memory	Data Memory
μ PD75402A	1920 \times 8 (mask ROM)	64 \times 4 (RAM)
μ PD75P402	1920 \times 8 (one-time PROM)	64 \times 4 (RAM)

Remarks In this manual, the μ PD75402A is described as the representative model unless there are particular differences in functions. If using it as the μ PD75P402 manual, the μ PD75402A, should be read as the μ PD75P402.

1.1 OUTLINE OF FUNCTIONS

Item		Description
Number of basic instructions		37
Instruction execution time		<ul style="list-style-type: none"> 0.95 μs, 1.91 μs, 15.3 μs (at 4.19 MHz operation) selectable between 3 levels
Built-in memory	Program memory	1920 \times 8 bits (μ PD75402A: Mask ROM, μ PD75P402: One-time PROM)
	Data memory	64 \times 4 bits (RAM)
General register		4 bits \times 4, or 8 bits \times 2 (memory mapping)
Accumulators		3 accumulators to suit manipulation data length <ul style="list-style-type: none"> Bit accumulator (CY), 4-bit accumulator (A), 8-bit accumulator (XA)
I/O line		Total 22 lines <ul style="list-style-type: none"> CMOS input port : 6 lines CMOS input/output port (LED direct drivable 8 lines) : 12 lines N-ch open-drain input/ output port (LED direct drivable) : 4 lines
Pull-up resistor		<ul style="list-style-type: none"> Pull-up resistor built-in control possible by software : 16 lines Pull-up resistor built-in control possible by mask option (μPD75402A only) : 4 lines
Clock output		<ul style="list-style-type: none"> 1.05 MHz, 524 kHz, 65.5 kHz (at 4.19 MHz operation) Applicable to remote control output
Timer/Counter		8-bit basic interval timer <ul style="list-style-type: none"> Reference time generation (1.95 ms, 31.3 ms : at 4.19 MHz operation) Watchdog timer applicable
Serial interface		<ul style="list-style-type: none"> 8 bits 2 transfer modes (clock synchronous 3-wire system mode/SBI mode)
Vectored interrupt		External: 1 line, internal : 2 lines
Test input		External: 1 line (For details, see CHAPTER 6 "INTERRUPT FUNCTIONS" .)
Standby		STOP mode/HALT mode
Instruction set		<ul style="list-style-type: none"> Bit manipulation instruction (set, clear, test, Boolean operation) 1-byte relative branch instruction 4-bit operation instruction (add, Boolean operation, compare) 4, 8-bit data transfer instruction
Package		<ul style="list-style-type: none"> 28-pin plastic DIP (600 mil) 28-pin plastic shrink DIP (400 mil) 44-pin plastic QFP (□ 10 mm)

1.2 ORDERING INFORMATION AND QUALITY GRADE

(1) Ordering Information

Ordering Code	Package	Program Memory
μ PD75402AC-xxx	28-pin plastic DIP (600 mil)	Mask ROM
μ PD75402ACT-xxx	28-pin plastic shrink DIP (400 mil)	
μ PD75402AGB-xxx-3B4	44-pin plastic QFP (□ 10mm)	
μ PD75P402C	28-pin plastic DIP (600 mil)	One-time PROM
μ PD75P402CT	28-pin plastic shrink DIP (400 mil)	
μ PD75P402GB-3B4	44-pin plastic QFP (□ 10mm)	

xxx: ROM code number

(2) Quality Grade

Standard

Please refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

1.3 DIFFERENCES BETWEEN μ PD75402A AND μ PD75402, 75P402

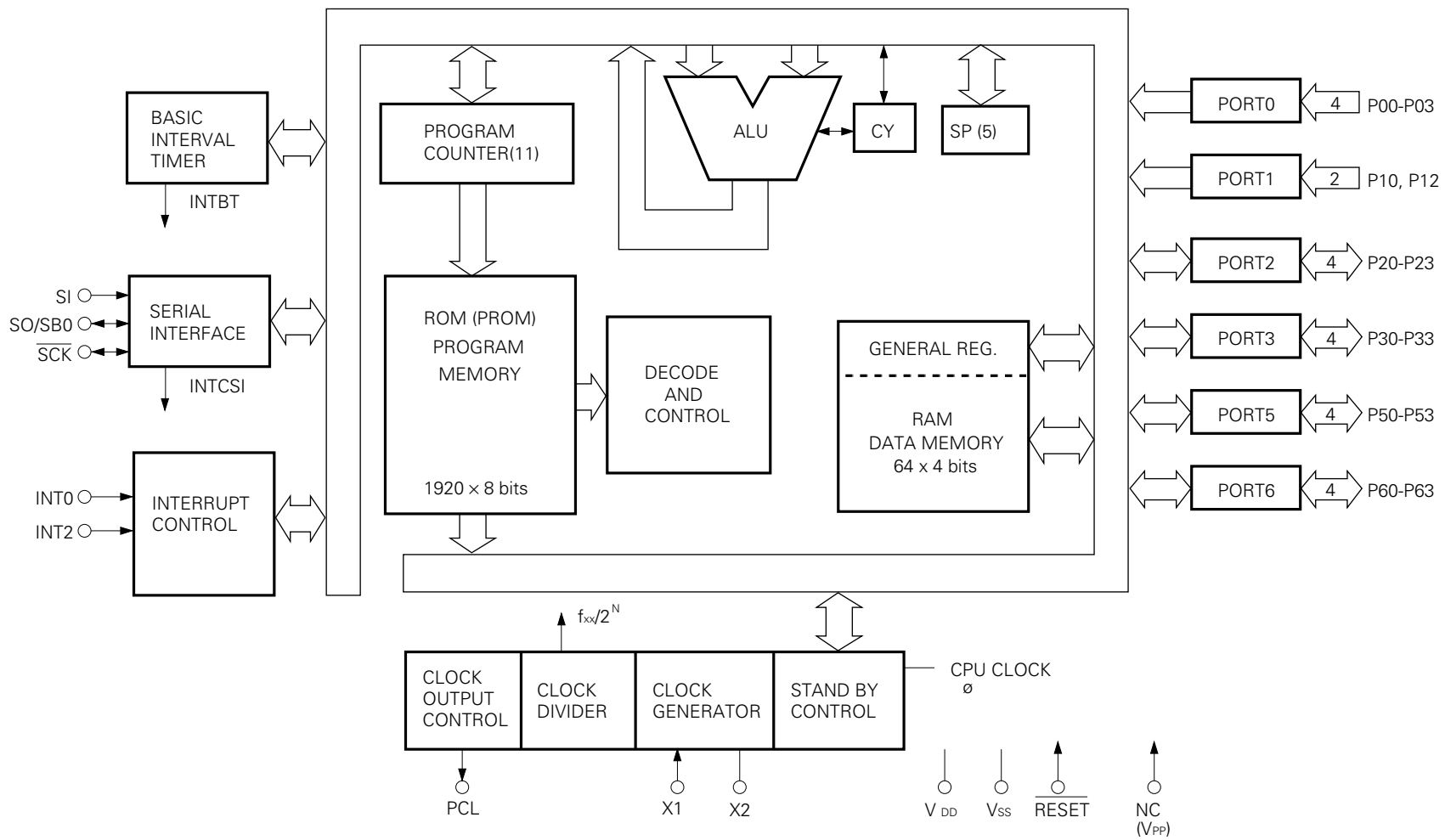
Table 1-1 shows the differences between the μ PD75402A and the μ PD75402, 75P402. Otherwise the μ PD75402A and the μ PD75402, 75P402 have the same functions and are pin-compatible.

Table 1-1 Differences Between μ PD75402A and μ PD75402, 75P402

Item		μ PD75402A	μ PD75402	μ PD75P402
ROM configuration		Mask ROM		One-time PROM
Instruction execution time		0.95, 1.91, 15.3 μ s (at 4.19 MHz operation)	1.91, 15.3 μ s* (at 4.19 MHz operation)	0.95, 1.91, 15.3 μ s (at 4.19 MHz operation)
Port 5's pull-up resistor		Designatable to be built in by mask option		Not available
Pin functions	1-pin (SDIP)	NC		V_{PP}
	30-pin (QFP)			
Supply voltage		2.7 to 6.0 V		5 V \pm 10%
Operating temperature range		-40 to +85°C		-10 to +70°C
Package		<ul style="list-style-type: none"> • 28-pin plastic DIP (600 mil) • 28-pin plastic shrink DIP (400 mil) • 44-pin plastic QFP (□ 10 mm) 		

* The μ PD75402A has shortened the μ PD75402's minimum instruction execution time to 0.95 μ s. For details, see 5.2.4 "Differences Between μ PD75402A And μ PD75402".

1.4 BLOCK DIAGRAM

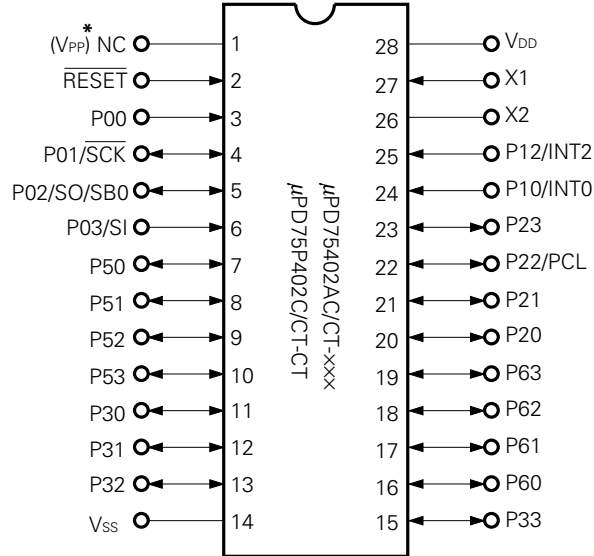


Remarks Parentheses for the μ PD75P402.

1.5 PIN CONFIGURATION

1.5.1 28-Pin Plastic Dip (600 mil), Shrink Dip (400 mil)

(1) Normal operating mode

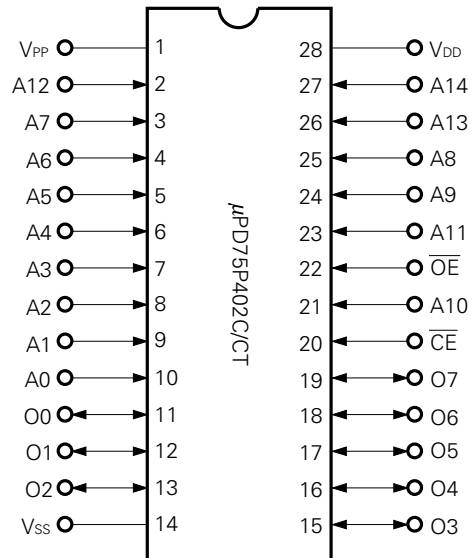


P00 to P03	: Port 0	SCK	: Serial clock input/output
P10, P12	: Port 1	SO/SB0	: Serial output/input/output
P20 to P23	: Port 2	SI	: Serial input
P30 to P33	: Port 3	PCL	: Clock output
P50 to P53	: Port 5	INT0	: External vectored interrupt input
P60 to P63	: Port 6	INT2	: External test input
		X1, X2	: Oscillator pin
		RESET	: Reset input
		VDD	: Power supply
		Vss	: Ground
		VPP	: Externally set to GND potential
		NC	: No connection

Remarks Parentheses for the μ PD75P402.

* If using the μ PD75P402 and the printed circuit board commonly in the μ PD75402A, the NC pin is to be set to the GND potential.

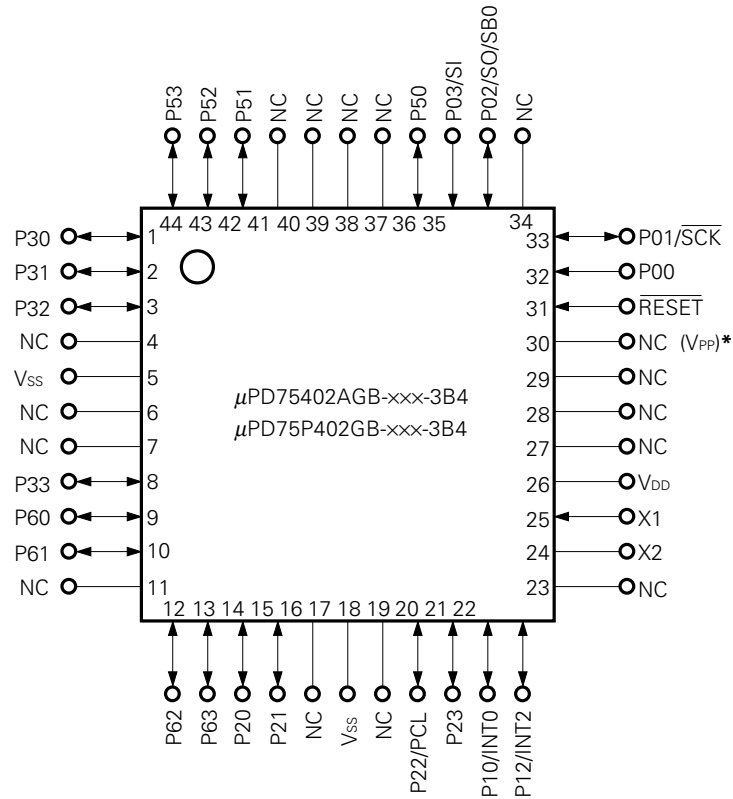
(2) PROM mode



- A0 to A14 : Address input
- O0 to O7 : Data input/output
- \overline{CE} : Chip enable input
- \overline{OE} : Output enable input
- V_{DD} : Power supply
- V_{PP} : Program power supply
- V_{SS} : Ground

1.5.2 44-Pin Plastic QFP (□ 10mm)

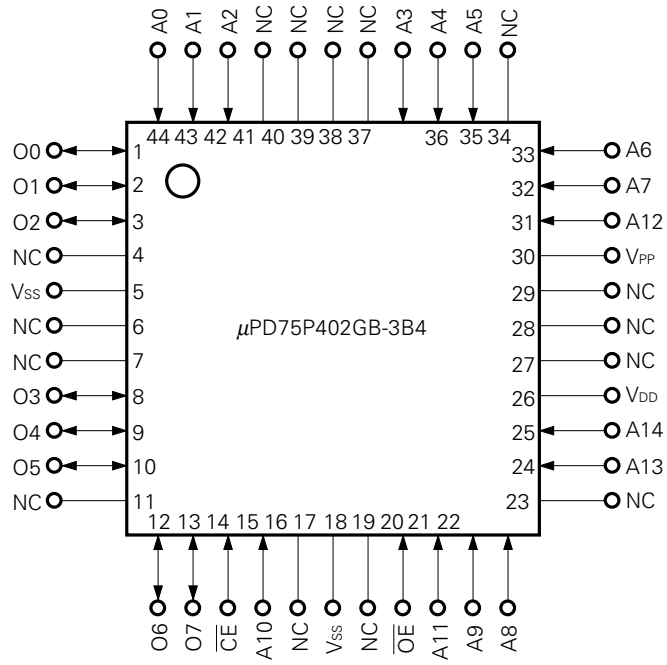
(1) Normal operating mode



Remarks Parentheses for the μ PD75P402.

* If using the μ PD75P402 and the printed circuit board commonly in the μ PD75402A, the NC pin of the 30-pin corresponding to the μ PD75P402's V_{PP} is to be set to the GND potential.

(2) PROM mode



CHAPTER 2. PIN FUNCTIONS

The μ PD75402A operates by the pin functions in the normal operating mode.

For the μ PD75P402's pin functions, the 2 modes of the normal operating mode (μ PD75402A mode) and the PROM mode are available.

The operating mode switches according to the V_{PP} pin level as shown in the table below.

V_{PP}	Operating Mode	
Low level (GND potential)	Normal operating mode	
High level (+5 V)	PROM mode	PROM read mode
High level (+12.5 V)		PROM write/verify mode

2.1 μ PD75402A PIN FUNCTION LIST

2.1.1 Port Pin List

Table 2-1 Port Pin List

Pin Name	Input/Output	Dual-Function Pin	Functions
P00	Input	————	A 4-bit input port (Port 0).
P01	Input/output	$\overline{\text{SCK}}$	For P01 to P03, it is designatable to build in the pull-up resistor by software in 3-bit units.
P02	Input/output	SO/SB0	
P03	Input	SI	
P10	Input	INT0	A 2-bit input port (Port 1). P10 is built in with the noise eliminator by the sampling clock.
P12		INT2	P12 is built in with the noise eliminator by analog delay. For P12, it is designatable to build in the pull-up resistor by software.
P20	Input/output	————	A 4-bit input/output port (Port 2).
P21		————	It is designatable to input/output in 4-bit units.
P22		PCL	It is designatable to build in the pull-up resistor by software in 4-bit units.
P23		————	
P30 to P33	Input/output	————	A programmable 4-bit input/output port (Port 3). It is designatable to input/output bit-wise. It is designatable to build in the pull-up resistor by software in 4-bit units. It is possible to drive the LED directly.
P50 to P53	Input/output	————	A 4-bit N-ch open drain input/output port (Port 5). It is designatable to input/output in 4-bit units. It is designatable to build in the pull-up resistor by mask option bit-wise. It is possible to drive the LED directly.
P60 to P63	Input/output	————	A 4-bit input/output port (Port 6). It is designatable to input/output in 4-bit units. It is designatable to build in the pull-up resistor by software in 4-bit units. It is possible to drive the LED directly.

- Remarks**
1. In the μ PD75402A, 8-bit input/output with 2 ports making up a pair is impossible.
 2. For the status of each pin at reset, see **CHAPTER 8 "RESET FUNCTION"**.

2.1.2 List of Pins Other Than Port Pins

Table 2-2 List of Pins Other than Port Pins

Pin Name	Input/Output	Dual-Function Pin	Functions
INT0	Input	P10	An edge-detected vectored interrupt request input pin (detected edge selectable by mode register). Built in with the noise eliminator by the sampling clock.
INT2	Input	P12	An edge detected external test input pin (rising edge detection).
SI	Input	P03	A serial data input pin.
SO	Input/output	P02/SB0	A serial data output pin.
$\overline{\text{SCK}}$	Input/output	P01	A serial clock input/output pin.
SB0	Input/output	P02/SO	A serial bus input/output pin.
PCL	Input/output	P22	A clock output pin.
X1, X2	Input	————	A system clock oscillation crystal/ ceramic resonator connection pin. If supplying the clock from the exterior, input to X1 and input the inverted phase to X2.
$\overline{\text{RESET}}$	Input	————	A system reset input pin. Built in with the noise eliminator by analog delay.
V _{DD}	————	————	A positive power supply pin.
V _{SS}	————	————	A GND potential pin.
NC*	————	————	No Connection

Remarks For the status of each pin at reset, see **CHAPTER 8 “RESET FUNCTION”**.

* If using the μ PD75P402 and the printed circuit board commonly, the NC pin should be connected directly to V_{SS}.

2.2 NORMAL OPERATING MODE

2.2.1 P00 to P03 (Port 0) \overline{SCK} , SO/SB0, SI Dual-Function Input
 P10, P12 (Port 1) INT0, INT2 Dual-Function Input

P00 to P03 are the 4-bit input port: Port 0's input pins. P10 and P12 are the 2-bit input port: Port 1's input pins.

Ports 0 and 1 also have the functions of the various control signal pins shown in Table 2-1 in addition to the functions as input ports. The status of each of Ports 0 and 1 is always inputtable irrespective of the dual-function pin operation.

Both Ports 0 and 1 have Schmitt-triggered input to prevent malfunction by noise. P10 is built in with the noise eliminator by the sampling clock and P12 is built in with the noise eliminator by analog delay.

Ports 0 and 1 allow to designate to build in the pull-up resistor respectively in 3-bit units (P01 to P03) and bit-wise (P12 only). Such designation is made using the pull-up resistor designation register (POGA). Neither P00 or P10 can be built in with the pull-up resistor.

Any of these pins assumes the input port mode at \overline{RESET} input.

Table 2-3 Port 0's, 1's Dual-Function Pins

Port 0	Dual-Function	Pin Port 1	Dual-Function Pin
P00	————	P10	INT0
P01	\overline{SCK}	P12	INT2
P02	SO/SB0		
P03	SI		

2.2.2 P20 to P23 (Port 2) PCL Dual-Function 3-State Input/Output
P30 to P33 (Port 3) 3-State Input/Output
P50 to P53 (Port 5) N-ch Open Drain Middle-Voltage (10 V) Input/Output
P60 to P63 (Port 6) 3-State Input/Output

The 4-bit input/output port with the output latch: Port 2's, 3's, 5's, 6's 4-bit input/output pins. Port 2 also shares the programmable clock output (PCL) function with P22 in addition to having the input/output port function. Port 5 has the N-ch open drain middle-voltage (10 V) output.

Port 3 allows to designate input/output bit-wise using the port mode register (PMGA). Ports 2, 5 and 6 allow to designate input/output in 4-bit units using the port mode register (PMGA, PMGB).

Ports 2, 3 and 6 allow to designate to build in the pull-up resistor by software in 4-bit units. Such designation is made using the pull-up resistor designation register (POGA).

The μ PD75402A's Port 5 allows to designate to build in the pull-up resistor by mask option bit-wise. The μ PD75P402's Port 5 cannot be built in with the pull-up resistor. Ports 3, 5 and 6 have large-current output and can drive the LED directly.

Ports 2, 3 and 6 turn input ports (output high impedance) at $\overline{\text{RESET}}$ input. Port 5 turns high level (if built in with the pull-up resistor) or high impedance. The content of the output latch turns indeterminate.

2.2.3 $\overline{\text{SCK}}$, SO/SB0, SI Port 0 Dual-Function 3-State Input/Output

A serial interface input/output pin. It operates according to the serial operating mode register (CSIM) setting. Each has Schmitt-triggered input.

The serial interface stops at $\overline{\text{RESET}}$ input and each turns into an input port.

2.2.4 INT0 Port 1 Dual-Function Input

An external interrupt request input pin. It is designatable for either of the 3 of rising edge detection, falling edge detection and rising and falling edge detection using the external interrupt mode register (IM0).

INT0 has Schmitt-triggered input and is built in with the noise eliminator by the sampling clock.

2.2.5 INT2 Port 1 Dual-Function Input

An external test input pin. The detected edge is fixed to the rising edge. It has Schmitt-triggered input and is built in with the noise eliminator by analog delay.

INT2 has asynchronous input. It accepts a signal having a certain high-level width irrespective of the CPU's operation clock if one is input.

2.2.6 PCL Port 2 Dual-Function Output

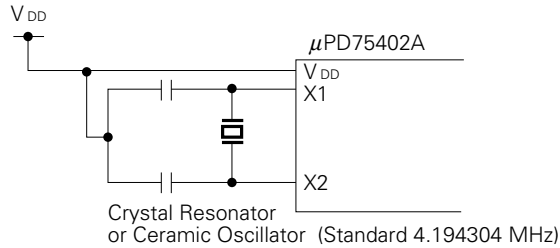
A programmable clock output pin. It is used to supply the clock to the peripheral LSI. The PCL output is also applicable to the remote control carrier signal.

The clock output function stops at $\overline{\text{RESET}}$ input and PCL turns into an input port (P22).

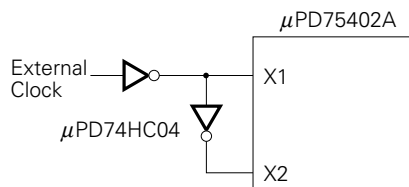
2.2.7 X1, X2 (Crystal)

The built-in clock oscillation crystal/ceramic input.
It is also possible to supply the clock from the exterior.

(a) Crystal/Ceramic Oscillation



(b) External Clock



2.2.8 $\overline{\text{RESET}}$ (Reset)

A low level active system reset input pin. It has Schmitt-triggered input and is built in with the noise eliminator by analog delay.

It has asynchronous input for $\overline{\text{RESET}}$. It accepts a signal having a certain low-level width irrespective of the CPU's operation clock if one is input and system reset is effected under priority over any other operation.

2.2.9 VDD

A positive power supply pin.

2.2.10 VSS

A GND potential pin.

2.3 PROM MODE

The PROM mode is designatable in the μ PD75P402 alone.

2.3.1 A0 to A14 (Address) Input

A 15-bit address input pin at PROM write/verify, read. As the PROM built into the μ PD75P402 has 2K bytes, it is addressed by the low-order 11 bits (A0 to A10). A11 to A14 should be fixed to the low level.

2.3.2 O0 to O7 (Data) Input/Output

An 8-bit data input/output pin at PROM write/verify, read.

2.3.3 $\overline{\text{CE}}$ (Chip Enable) Input

A chip enable signal input pin.

2.3.4 $\overline{\text{OE}}$ (Output Enable) Input

An output enable signal input pin.

2.3.5 V_{PP}

A high voltage application pin at PROM write/verify.
It must be connected to VSS during normal operation.

2.3.6 V_{DD}

A supply voltage application pin.

2.3.7 V_{SS}

A GND potential pin.

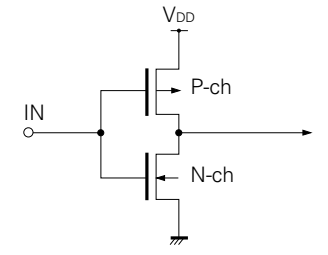
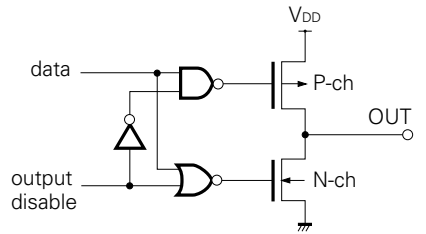
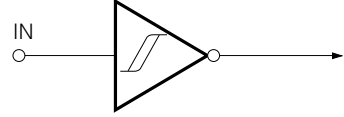
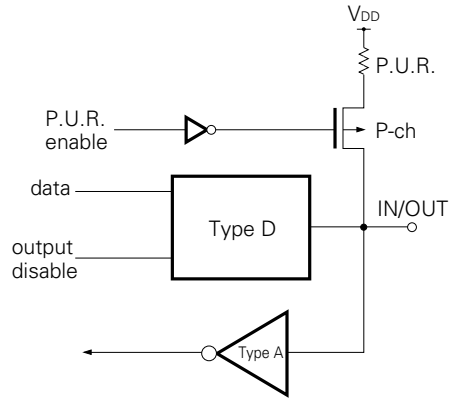
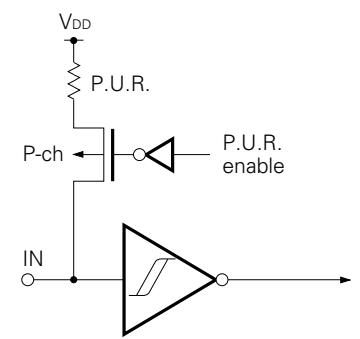
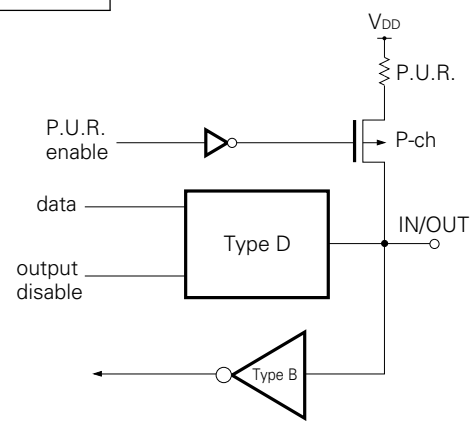
2.4 PIN INPUT/OUTPUT CIRCUITS

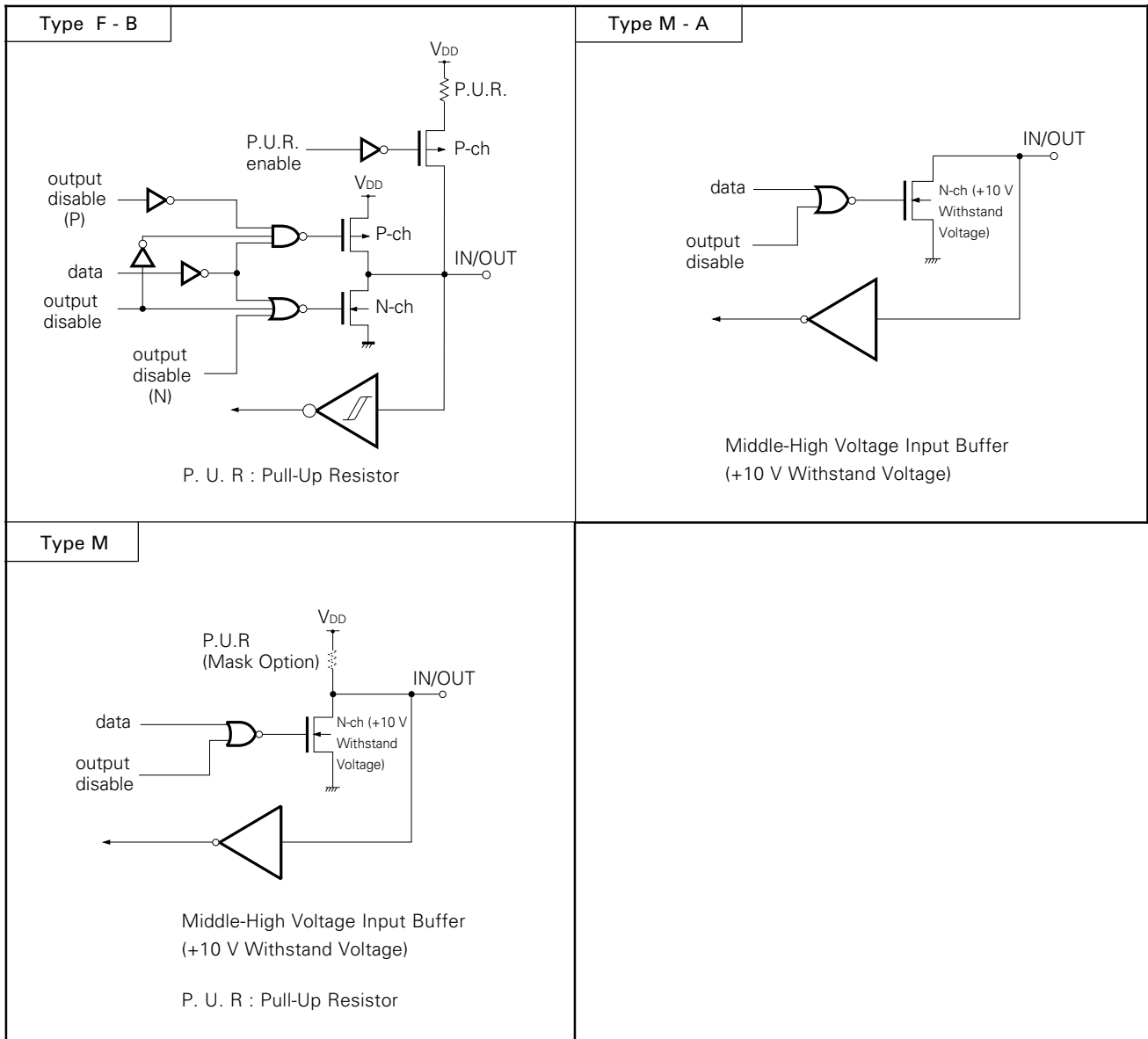
The input/output circuit of each pin is shown below in a partly simplified format.

Table 2-4 Pin Input/output Types

Pin	Input/Output Type	
	μ PD75402A	μ PD75P402
P00	ⓑ	
P01/SCK	ⓕ- A	
P02/SO/SB0	ⓕ- B	
P03/SI	ⓑ- C	
P10/INT0	ⓑ	
P12/INT2	ⓑ- C	
P20, P21, P23	E - B	
P22/PCL		
P30 to P33	E - B	
P50 to P53	M	M - A
P60 to P63	E - B	
$\overline{\text{RESET}}$	ⓑ	

Remarks A circle ○ indicates Schmitt-triggered input.

<p>Type A (for Types E - B)</p>  <p>An input buffer of the CMOS standard</p>	<p>Type D (for Type E - B, F - A, Y - D)</p>  <p>Push-pull output that can be turned output high impedance (P-ch, N-ch, both off)</p>
<p>Type B</p>  <p>Schmitt-triggered input having hysteresis characteristics</p>	<p>Type E - B</p>  <p>P. U. R : Pull-Up Resistor</p>
<p>Type B - C</p>  <p>P. U. R : Pull-Up Resistor</p>	<p>Type F - A</p>  <p>P. U. R : Pull-Up Resistor</p>



2.5 UNUSED PIN TREATMENT

Pin	Recommended Connection Method
P00	Connect to V _{SS} .
P01 to P03	• With pull-up resistor Connect to V _{DD} .
P10 and P12	• Without pull-up resistor Connect to V _{SS} or V _{DD} .
P20 to P23	• With pull-up resistor Input status: Connect to V _{DD} .
P30 to P33	Output status: Leave open.
P50 to P53	• Without pull-up resistor Input status: Connect to V _{SS} to V _{DD} .
P60 to P63	Output status: Leave open.
NC	Leave open or connect directly to V _{SS} .*

* If using the μ PD75P402 and the printed circuit board commonly, the NC pins should be connected directly to V_{SS}.

2.6 NOTES ON USE OF P00 PIN AND $\overline{\text{RESET}}$ PIN

The P00 and $\overline{\text{RESET}}$ pins are provided with the test mode setting function of test mode (for IC test) which tests the internal operation of the μ PD75402A in addition to the functions described in 2.2.1 and 2.2.8.

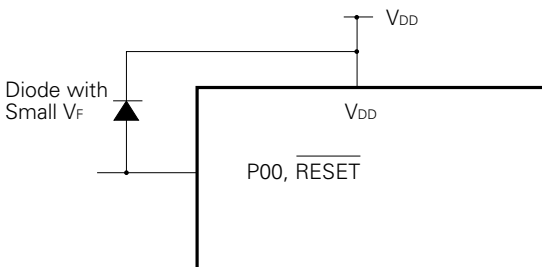
When a voltage exceeding V_{DD} is applied to either of these pins, the test mode is set. Consequently, if noise exceeding V_{DD} is added even in a normal operation, the test mode is set and the normal operation may not be continued.

For example, when wires from the P00 pin or $\overline{\text{RESET}}$ pin are long, inter-wiring noise may be added to these pins and the pin voltage may exceed V_{DD}, resulting in a misoperation.

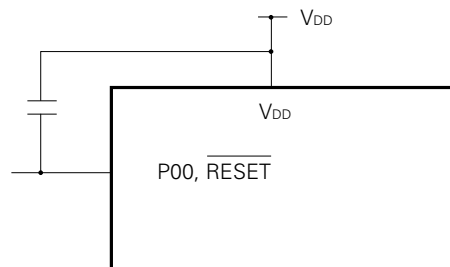
Therefore, wiring should be carried out so that inter-wiring noise is suppressed as far as possible.

If it is impossible to suppress the noise, noise prevention measures by means of external parts should be taken as shown below.

- o Insert diode with small V_F (less than 0.3 V) between V_{DD} and P00/ $\overline{\text{RESET}}$ pin.



- o Insert capacitor between V_{DD} and P00/ $\overline{\text{RESET}}$ pin.



CHAPTER 3. FEATURES OF ARCHITECTURE AND MEMORY MAP

The μ PD75402A's architecture is a subset of the 75X architecture. Its features are outlined below.

3.1 DATA MEMORY BANK CONFIGURATION AND ADDRESSING MODES

3.1.1 Data Memory Bank Configuration

The μ PD75402A's data memory space has a bank configuration. Addresses 000H to 03FH of Bank 0 are a data area as shown in Table 3-1 and are built in with a static RAM (64×4 bits). Addresses F80H to FFFH of Bank 15 are a peripheral hardware area and are built in with the input/output port, serial interface, etc. To address this data memory space of a 12-bit address, the low-order 8-bit address is specified directly or indirectly by an instruction. The high-order 4-bit address is determined by the memory bank (MB) to be accessed.

The μ PD75402A is built in with only Memory bank 0 and 15 and does not require bank switching unlike other products of the 75X series. The memory bank to be accessed is determined by the addressing mode and the address to be specified (see Tables 3-1 and 3-2).

Table 3-1 Data Memory Configuration and Address Range in Each Addressing Mode

Address	Data Memory	Addressing Mode	mem mem. bit	@ HL	Stack Addressing	fmem. bit
000H	↑ General Register Area ↓			/ / / / / / / /		
003H						
020H	Data Memory Static RAM (Memory Bank 0) ↓ Stack Area ↑			/ / / / / / / /	/ / / / / / / /	
03FH						
	Not built in.					
F80H	Peripheral Hardware Area (Memory Bank 15) ↓ ↑					
FB0H						
FBFH						
FF0H						/ / / / / / / /
FFFH						/ / / / / / / /

Table 3-2 Addressing Mode List

Addressing Mode	Notation	Specified Address
1-bit direct addressing	mem. bit	The bit indicated by bit of the address indicated by mem. However: { Memory bank 0 is accessed if mem = 00H to 3FH. Memory bank 15 is accessed if mem = 80H to FFH.
4-bit direct addressing	mem	The address indicated by mem. However: { Memory bank 0 is accessed if mem = 00H to 3FH. Memory bank 15 is accessed if mem = 80H to FFH.
8-bit direct addressing		The address indicated by mem (mem: Even address). However: { Memory bank 0 is accessed if mem = 00H to 3EH. Memory bank 15 is accessed if mem = 80H to FEH.
4-bit register indirect addressing	@HL	The address indicated by the content of HL of Memory bank 0. If HL = 00H to 3FH, however.
Bit manipulation addressing	fmem. bit	The bit indicated by bit of the address indicated by fmem of Memory bank 15. If the following, however: { fmem = FB0H to FBFH (interrupt, etc. hardware) fmem = FF0H to FFFH (input/output port)
Stack addressing	-	The address indicated by SP of Memory bank 0. Limited to 20H to 3FH, however.

3.1.2 Data Memory Addressing Modes

In the μ PD75402A, the 6 types of addressing modes listed on Table 3-2 are available for the data memory space for efficient addressing per the bit length of the data to be processed.

Also in the μ PD75402A, the memory bank to be accessed is fixed by the addressing mode unlike in other products of the 75X series. So programming is possible without caring about memory bank switching.

(1) 1-bit direct addressing (mem.bit)

An addressing mode to specify each bit of the whole data memory space directly by the instruction's operand.

The specified memory bank (MB) is MB = 0 if the address specified by the operand is 00H to 3FH and MB = 15 if it is 80H to FFH. Consequently, all the bits in both the static RAM area of 000H to 03FH and the peripheral hardware area of FF0H to FFFH are addressable. In the peripheral hardware area, however, the bits capable of 1-bit manipulation are limited (see Table 3-2).

This addressing mode is applied to the 4 instructions of the bit set and reset instructions (SET1, CLR1) and test instructions (SKT, SKF).

Example Set FLAG1, reset FLAG2 and test whether FLAG3 is 0 or not.

```

FLAG1 EQU    03FH.1    ; Address 3FH, bit 1
FLAG2 EQU    027H.2    ; Address 27H, bit 2
FLAG3 EQU    017H.0    ; Address 17H, bit 0

      SET1    FLAG1     ; FLAG ← 1
      CLR1    FLAG2     ; FLAG ← 0
      SKF     FLAG3     ; FLAG = 0?

```

(2) 4-bit direct addressing (mem)

An addressing mode to specify the whole data memory space directly by the instruction's operand per 4 bits.

The specified memory bank (MB) is MB = 0 if the address specified by the operand is 00H to 3FH and MB = 15 if it is 80H to FFH. Consequently, both the static RAM area of 000H to 03FH and the peripheral hardware area of FF0H to FFFH are addressable.

This addressing mode is applied to the MOV, XCH, INCS, IN, and OUT instructions.

Example 1. Input Port 2 and store it in "DATA1".

```

DATA1 EQU    2FH        ; "DATA1" is at address 2FH
      IN     A, PORT 2  ; A ← Port 2
      MOV    DATA1, A  ; (DATA1) ← A

```

2. Output the data of "BUFF" to Port 5.

```

BUFF EQU    01AH       ; "BUFF" is at address 01AH
      MOV    A, BUFF   ; A ← (BUFF)
      OUT    PORT 5, A ; PORT 5 ← A

```

(3) 8-bit direct addressing (mem)

An addressing mode to specify the whole data memory space directly by the instruction's operand per 8 bits.

The specified memory bank (MB) is MB = 0 if the address specified by the operand is 00H to 3EH and MB = 15 if it is 80H to FEH. Consequently, both the static RAM area of 000H to 03FH and the peripheral hardware area of FF0H to FFFH are addressable. In the peripheral hardware area, however, the addresses capable of 8-bit manipulation are limited. (See Table 3-4).

This addressing mode is applied to the MOV and XCH instructions.

Example 1. Store the serial interface shift register's (SIO) 8-bit data at addresses 20H and 21H.

```

DATA      EQU      020H
          MOV      XA, SIO      ; XA ← SIO
          MOV      DATA, XA    ; (21H) ← X, (20H) ← A
    
```

2. Take the 8-bit data input to the SIO into the XA register pair as well as setting the transfer data stored in the XA register pair.

```

XCH      XA, SIO      ; XA ↔ SIO
    
```

(4) 4-bit register indirect addressing (@HL)

An addressing mode to specify the data memory space indirectly according to the content of the HL register per 4 bits.

The memory bank (MB) addressed in this addressing mode is fixed to 0. Consequently, the static RAM area of 000H to 03FH alone is addressable. The peripheral hardware area is not addressable. Data in the range of 00H to 3FH should be set in the HL register pair.

This addressing mode is applied widely to data transfer, operation, compare, etc.

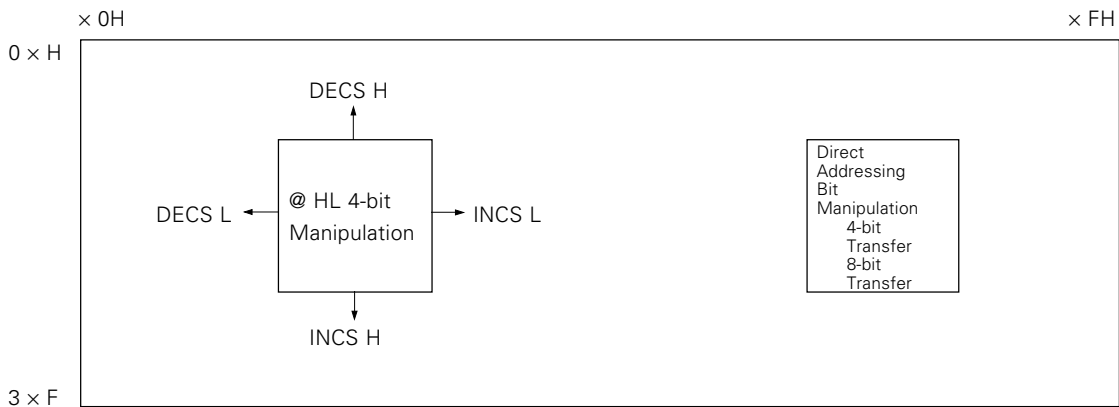
If combined with the HL register pair's increase or decrease instruction (INCS, DECS), meanwhile, this addressing mode allows the data memory space address to be updated freely.

Example Turn all the contents of 20H to 2FH into FH.

```

          MOV      HL, #2FH
          MOV      A, #0FH      ; A ← FH
LOOP:    MOV      @HL, A      ; (HL) ← A
          DECS    L
          BR      LOOP
    
```

Fig. 3-1 Static RAM Address Updating Method



(5) Specific address bit manipulation addressing (fmem. bit)

An addressing mode to specify each bit of the input/output port, interrupt, etc. flag, etc. of the peripheral hardware directly by the instruction's operand. Consequently, the data memory addresses to which this addressing mode is applied are FB0H to FBFH, FF0H to FFFH.

While the 1-bit direct addressing mode (mem.bit) is applicable only to the bit set/reset/test instructions, this addressing mode enables multifarious bit manipulation such as the Boolean operation by the AND1, OR1 and XOR1 instructions, test and reset by the SKTCLR instruction in addition to them.

Example 1. Test the basic interval timer interrupt request flag (IRQBT) and, if set, clear IRQBT and reset the P63 pin level.

```
SKTCLR   IRQBT       ; IRQBT = 1?
BR       NO          ; NO
CLR1     PORT 6. 3   ; YES
```

2. If P30 and p61 both 1, reset P53.



```
(i) SET1   CY          ; CY ← 1
   AND1   CY, PORT3. 0 ; CY∧P30
   AND1   CY, PORT6. 1 ; CY∧P61
   SKT    CY          ; CY = 1?
   BR     SETP
   CLR1   PORT5. 3     ; P53 ← 0
```

```
·
·
SETP: SET1 PORT5. 3     ; P53 ← 1
```

```
(ii) SKT   PORT3. 0    ; P30 = 1?
   BR     SETP
   SKT    PORT6. 1    ; P61 = 1?
   BR     SETP
   CLR1   PORT5. 3    ; P53 ← 0
```

```
·
·
SETP: SET1 PORT5. 3     ; P53 ← 1
```

(6) Stack addressing

This addressing mode is for the saving/restoring operation during the interrupting process, subroutine process. The data memory is addressed indirectly according to the content of the stack pointer (SP : 8 bits).

The memory bank (MB) addressed in this addressing mode is fixed to 0. Also as the stack pointer's high-order 3 bits are fixed to 001, the addressable area is limited to 020H to 03FH.

This addressing mode is also applied at register save/restore by the PUSH, POP instruction in addition to the interrupting process, subroutine process.

Note The Evachip packaged on the board for evaluation can address the whole area of Memory bank 0 in this addressing mode unlike in the μ PD75402A. To eliminate such a difference during the evaluation, a value not to access beyond the range of 20H to 3FH should be set in the stack pointer.

Example 1. Save/restore the register during the subroutine process.

```

SUB:   PUSH    XA
       PUSH    HL
       .
       .
       .
       POP     HL
       POP     XA
       RET
    
```

2. Transfer the content of the HL register pair to the XA register pair.

```

PUSH    HL
POP     XA    ; XA ← HL
    
```

3. Branch to the address indicated by the (X AHL) register.

```

PUSH    HL
PUSH    XA
RET     ; Branch to address XAHL
    
```

3.2 MEMORY-MAPPED I/O

The μ PD75402A adopts memory-mapped I/O to map such peripheral hardware as the input/output port, serial interface at addresses F80H to FFFH in the data memory space shown in Table 3-1. As a result, there is no special instruction to control the peripheral hardware; the peripheral hardware is controlled wholly by memory manipulation instructions (some hardware control mnemonics are available to make the program easy to understand).

Table 3-3 shows the addressing modes available when operating the peripheral hardware.

Table 3-3 Applicable Addressing Modes at Peripheral Hardware Operation

	Applicable Addressing Mode	Applicable Hardware
Bit manipulation	Specify the bit to be manipulated by the direct addressing mem.bit.	All the hardware capable of bit manipulation
	Specify the bit to be manipulated by the direct addressing fmem.bit.	IE _{xxx} , IRQ _{xxx} , PORTn.x
4-bit manipulation	Specify the address to be manipulated by the direct addressing mem.	All the hardware capable of 4-bit manipulation.
8-bit manipulation	Specify the address to be manipulated by the direct addressing mem. Mem is an even address, however.	All the hardware capable of 8-bit manipulation.

Table 3-4 summarizes the μ PD75402A's I/O map. The items shown in this table have the following meaning.

- Symbol: A name to indicate the address of the built-in hardware. It can be described in the instruction's operand column. IME is excepted, however.
- Number of manipulatable bits: The number of applicable processing bits when operating the relevant hardware. Such symbols as R/W, indicate whether the relevant hardware is readable/writable or not.
R/W : Readable/Writable
R : Readable
W : Writable
- Bit manipulation addressing: The applicable bit manipulation addressing when bit manipulating the relevant hardware.

Table 3-4 μ PD75402A I/O Map (1/2)

Address	Hardware Name (Symbol)				No. of Manipulatable Addressing			Bit Manipulation	Remarks
	b3	b2	b1	b0	1 Bit	4 Bits	8 Bits		
F80H	Stack pointer (SP)				—	—	W		Bit 0 is fixed to 0.
					—	—			
F85H	Basic interval timer mode register (BTM)				—	W	—		11 must always be written in bit 1, 0.
F86H	Basic interval timer (BT)				—	—	R		
					—	—			

FB2H	(IME)				—	—	—		Manipulation by EI, DI instruction
FB3H	Processor clock control register (PCC)				—	W	—		
FB4H	INT0 mode register (IM0)				—	W	—	fmem. bit	Bit 2 is fixed to 0.
FB8H	0	0	IEBT	IRQBT	R/W	R/W	—		
FBDH	0	0	IECSI	IRQCSI	R/W	R/W	—		
FBEH	0	0	IE0	IRQ0	R/W	R/W	—		
FBFH	0	0	IE2	IRQ2	R/W	R/W	—		

FD0H	Clock output mode register (CLOM)				—	W	—		
FDCH	Pull-up resistor specify register Group A (POGA)				—	—	W		
					—	—			

- Remarks**
1. IE_{xxx} is an interrupt enable flag.
 2. IRQ_{xxx} is an interrupt request flag.

Table 3-4 μ PD75402A I/O Map (2/2)

Address	Hardware Name (Symbol)				No. of Manipulatable Addressing			Bit Manipulation	Remarks
	b3	b2	b1	b0	1 Bit	4 Bits	8 Bits		
FE0H	Serial operation mode register (CSIM)				—	—	W	mem. bit	0 must always be written in bit 0.
FE1H	CSIE	COI	WUP	0	* 1	—			
FE2H	CMDDD	RELD	CMDT	RELT	* 2	—	—	mem. bit	Bit manipulation only is possible for all the bits.
FE3H	SBI control register (SBIC)				* 3	—			
FE4H	Serial I/O shift register (SIO)				—	—	R/W		
					—	—			
FE6H	Slave address register (SVA)				—	—	W		11000 must always be written in the high-order 5 bits.
					—	—			
FE8H	Port mode register Group A (PMGA)				—	—	W		
					—	—			
FECH	Port mode register Group B (PMGB)				—	—	W		
					—	—			

FF0H	Port 0 (PORT 0)	R	R	—	fmem. bit	Bits 3 and 1 are fixed to 0.
FF1H	Port 1 (PORT 1)	R	R	—		
FF2H	Port 2 (PORT 2)	R/W	R/W	—		
FF3H	Port 3 (PORT 3)	R/W	R/W	—		
FF5H	Port 5 (PORT 5)	R/W	R/W	—		
FF6H	Port 6 (PORT 6)	R/W	R/W	—		

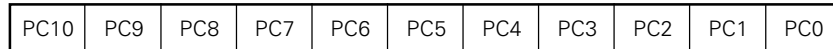
- * 1. Bits 3 and 1: W; bit 2: R.
 2. Bits 3 and 2: R; bits 1 and 0: W.
 3. Bits 3 and 1: R/W; bit 2: R; bit 0: W.

CHAPTER 4. INTERNAL CPU FUNCTIONS

4.1 PROGRAM COUNTER (PC) 11 BITS

An 11-bit binary counter to hold the program memory address information.

Fig. 4-1 Program Counter Configuration



The program counter operates as follows.

- Normal operation
The content is incremented automatically according to the number of bytes of the instruction every time one is executed.
- Branch instruction (BR, BR CB) execution
The immediate data indicating the address of the destination of branching is set in the PC.
- Subroutine call instruction (CALLF) execution and vector interrupt
The content of the PC at that time is saved in the stack memory and then the address of each destination of branching is set in the PC.
- Return instruction (RET, RETS, RETI) execution
The content of the stack memory is set in the PC.
- RESET input
The low-order 3 bits of the program memory's address 000H is set in PC10 to PC8 and the content of address 001H is set in PC7 to PC0 and then initialized. It is possible to start the program from any address.

4.2 PROGRAM MEMORY (ROM) 1,920 WORDS × 8 BITS

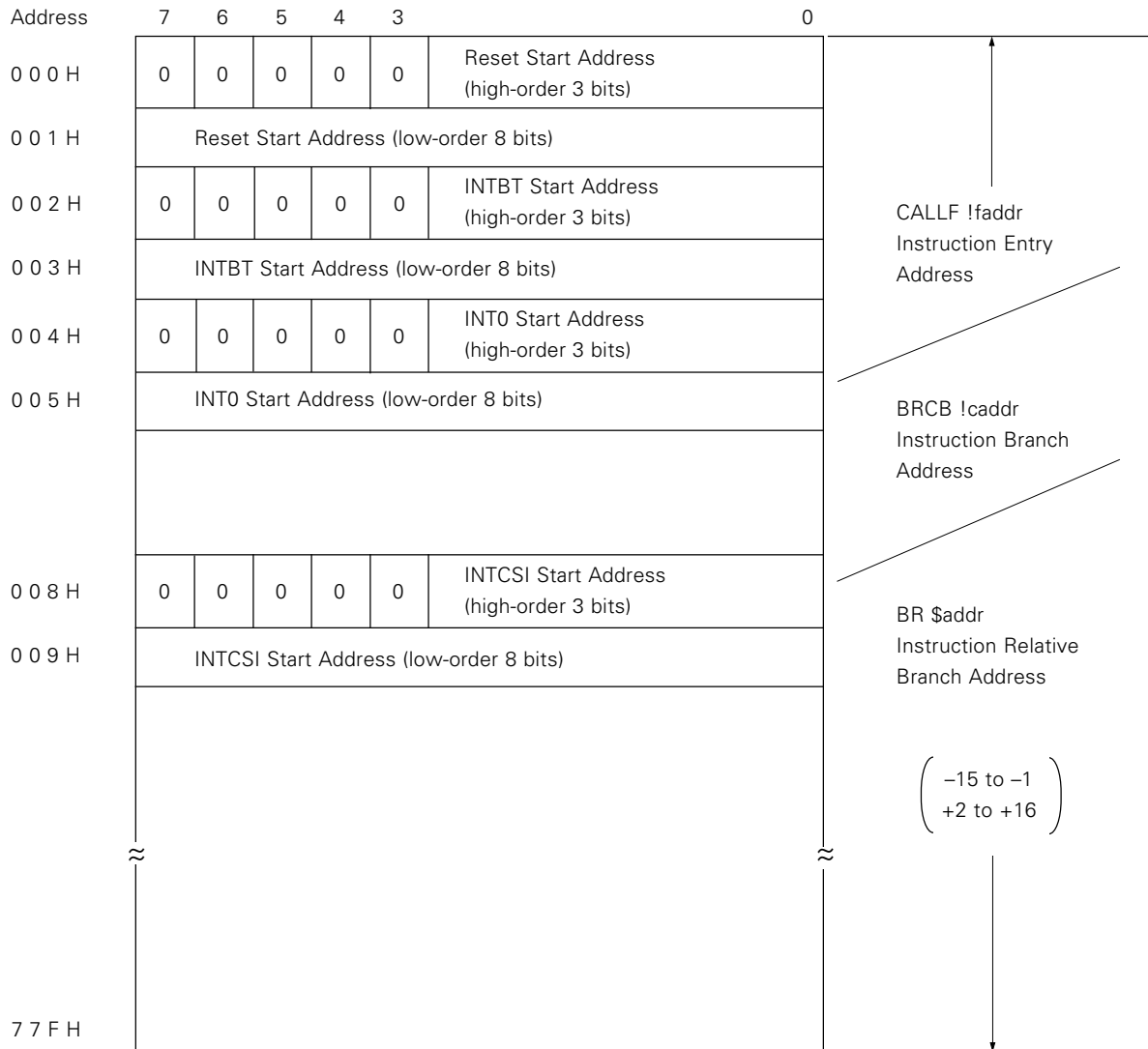
A mask programmable ROM of a 1,920-word × 8-bit configuration. It stores the program, table data, etc.

The program memory is addressed by the program counter. It is also possible to read the table data in the ROM by the table refer instruction (MOVT).

It is possible to branch to any area of the program memory by the branch instruction, subroutine call instruction (see Fig. 4-2). With the relative branch instruction (BR \$addr), it is possible to branch to the range of (-15 to -1, +2 to +16) from the address indicated by the PC after the instruction execution.

The program memory's addresses cover 000H to 77FH and the addresses shown below are assigned specially. All the areas excluding 000H and 001H are available as the normal program memory.

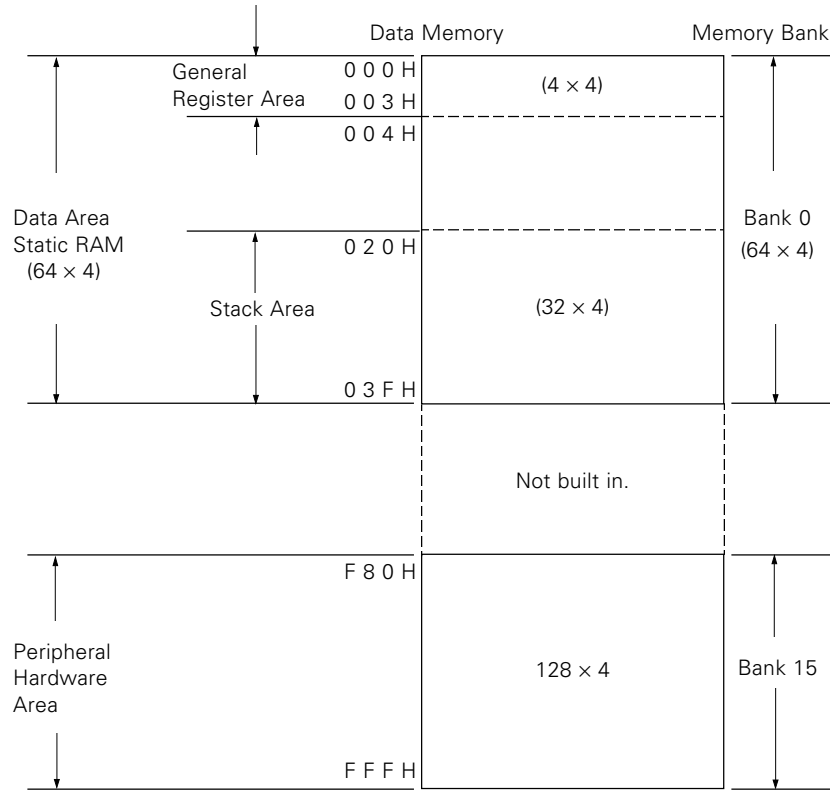
Fig. 4-2 Program Memory Map



4.3 DATA MEMORY (RAM)

The data memory consists of the data and peripheral hardware areas as shown in Fig. 4-3.

Fig. 4-3 Data Memory Map



(1) Data area

The μ PD75402A's data area consists of the static RAM (64 words × 4 bits). The data area is used to store processing data and is operated by the memory manipulation instruction.

The static RAM is mapped to Memory bank 0 by 64 × 4 bits. While Bank 0 is mapped as the data area, it is also available as the general register area (000H to 003H) and the stack area (020H to 03FH).

In the static RAM, 1 address consists of 4 bits. It is possible either to operate per 8 bits by the 8-bit memory manipulation instruction or to operate per bit by the bit manipulation instruction, however. In the 8-bit manipulation instruction, an even address should be specified.

• General register area

Operation is possible either by the general register manipulation instruction or by the memory manipulation instruction. Up to four 4-bit registers are available. Of the 4 general registers, that part which is not used in the program is available as the data area (see 4.4 "GENERAL REGISTER").

• Stack area

The stack area is set by an instruction and is available as the save area at the subroutine or interrupting process execution (see 4.6 "STACK POINTER"). In that case, the stack area is at the static RAM's addresses 020H to 03FH.

(2) Peripheral hardware area

The peripheral hardware area is mapped to memory bank 15's addresses F80H to FFFH.

The operation is performed by the memory manipulation instruction just as in the static RAM. In the peripheral hardware, however, the operable bit unit differs from one address to another. It is impossible to access an address to which the peripheral hardware is not assigned since the data memory is not built in. (See Table 3-4 " μ PD75402A I/O Map".)

Note The static RAM is indefinite at reset. Therefore, it should be initialized to zero at the beginning of the program (RAM clear). This must be carried out for sure to avoid unexpected bugs.

Example Clear all the areas (00H to 3FH) of the static RAM (FFH remains in the HL register, however).

```
                MOV     HL, #3FH
                MOV     A, #0H
LOOP:          MOV     @HL, A: Clear (0) 00H to 3FH
                DECS   L
                BR     LOOP
                DECS   H
                BR     LOOP
```

4.4 GENERAL REGISTER 4 × 4 BITS

The general register is assigned to a specific address of the data memory. There are four 4-bit registers (H, L, X, A).

While each general register is operated per 4 bits, HL and XA make up register pairs, each of which is operated per 8 bits. The HL register pair is available as the data pointer to indirectly address the memory.

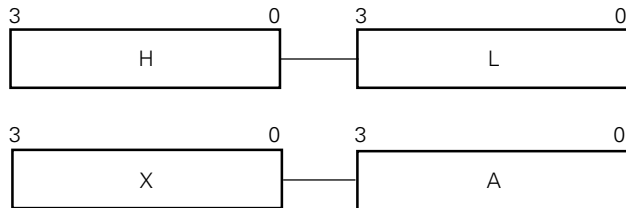
The general register area can be addressed and accessed as an ordinary RAM regardless of whether it is used as a register or not.

Fig. 4-4 General Register Configuration

X	0 1 H	A	0 0 H
H	0 3 H	L	0 2 H

Remarks The figure in the lower right corner is the assigned data memory address.

Fig. 4-5 Register Pair Configuration

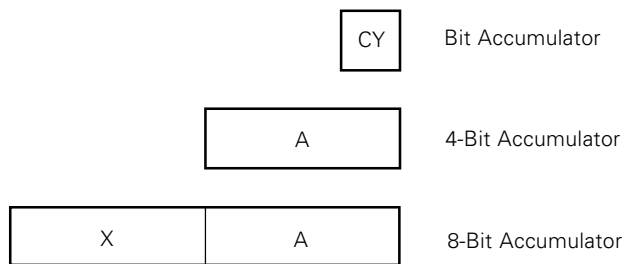


4.5 ACCUMULATOR

In the μ PD75402A, the A register and the XA register pair function as accumulators. The 4-bit data process instruction is executed mainly by the A register and the 8-bit data process instruction is executed mainly by the XA register pair.

In the bit manipulation instruction, the carry flag (CY) functions as the bit accumulator.

Fig. 4-6 Accumulators



4.6 STACK POINTER (SP) 8 BITS

The μ PD75402A uses a static RAM as the stack memory (LIFO format). The 8-bit register holding the top address information of such a stack memory area is the stack pointer (SP). Fig. 4-7 shows its format.

As the SP's high-order 3 bits are fixed to 001, the stack area is at the static RAM's addresses 020H to 03FH.

The SP is decremented before write (save) in the stack memory and is incremented after read (restore) from the stack memory. Figs. 4-8 and 4-9 show the data saved and restored by each stacking operation.

The SP sets the initial value by the 8-bit data transfer instruction to determine the stack area. It is impossible to read the content of the SP.

Zero is always written in the SP's bit 0.

It is recommended to initialize by writing 40H in the SP and use the built-in RAM's maximum address (03FH) and beyond as the stack area.

The content of the SP turns indeterminate at $\overline{\text{RESET}}$ input, so it must be initialized to your desired value by the program initialization routine.

Note The SP of the Evachip packaged on the board for evaluation can address all the areas of 000H to 0FFH unlike in the μ PD75402A. To eliminate such a difference at evaluation, the SP should be set not to access beyond the range of 020H to 03FH.

Example SP initialize

```
MOV      XA, #40H
MOV      SP, XA      ; SP ← 40H
```

Fig. 4-7 Stack Pointer Configuration

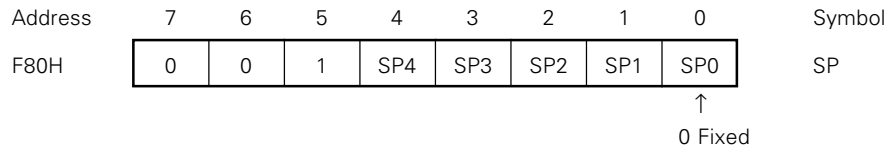


Fig. 4-8 Data Saved to Stack Memory

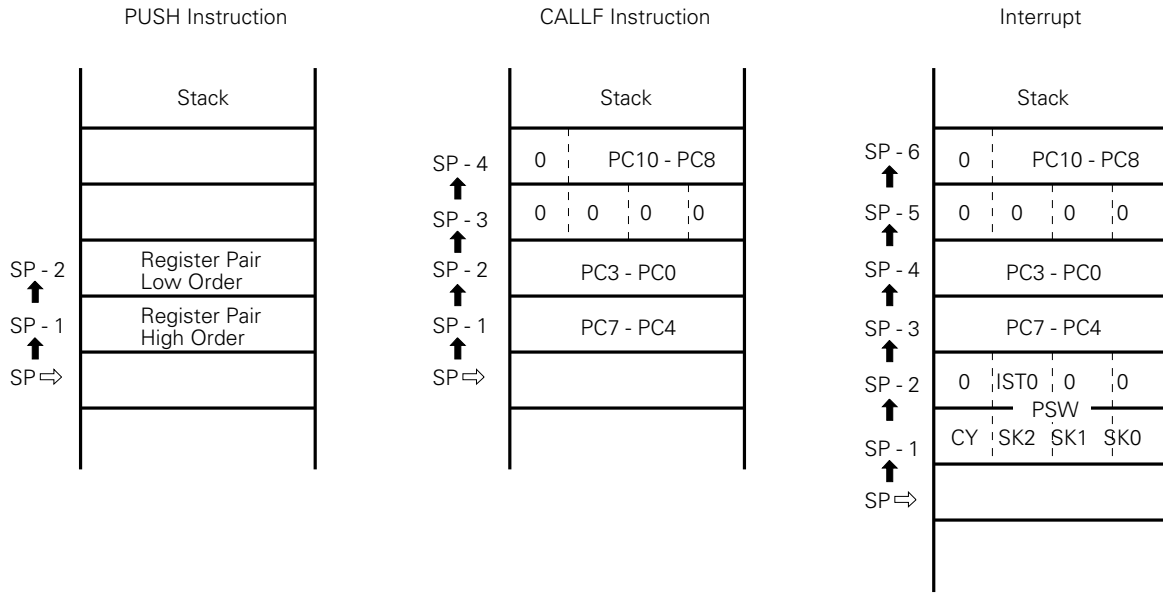
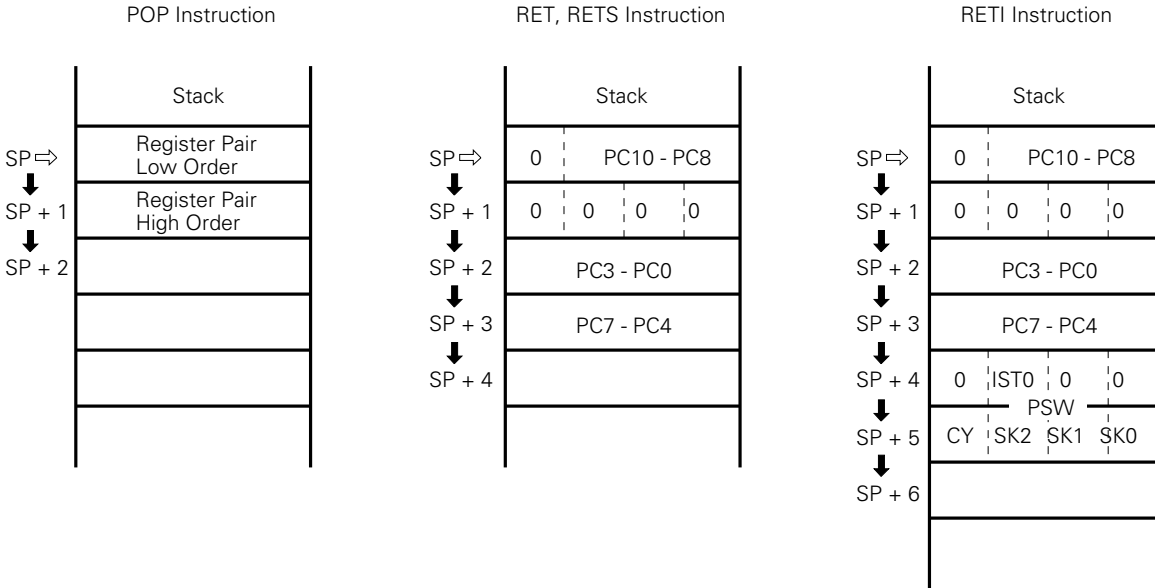


Fig. 4-9 Data Restored from Stack Memory

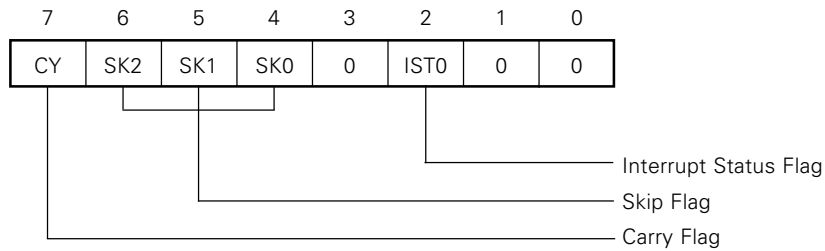


4.7 PROGRAM STATUS WORD (PSW) 8 BITS

The program status word (PSW) consists of various flags concerning closely the processor operation. Fig. 4-10 shows its configuration.

Saved to the stack memory per 8 bits at the interrupt acceptance and restored from the stack memory per 8 bits at the RETI instruction execution (see Figs. 4-8 and 4-9).

Fig. 4-10 Program Status Word Configuration



The PSW is not assigned to the data memory space. So it is impossible to operate each flag by the memory manipulation instruction. The carry flag (CY) alone is operable by a dedicated instruction, however.

IST0 turns 0 and SK0 to SK2 and CY turn indeterminate at RESET input.

(1) Carry flag (CY)

The carry flag is a 1-bit flag to store the overflow occurrence information at the operation instruction with carry (ADDC) execution.

The carry flag also has the function of the bit accumulator, so it can perform Boolean operation with the bit address specify data memory and can store its results.

The carry flag is operated by a dedicated instruction irrespective of the other PSW bits.

The carry flag turns indeterminate with the RESET input.

Table 4-1 Carry Flag Manipulation Instructions

	Instruction (Mnemonic)	Carry Flag Operation, Process
Carry flag manipulation dedicated	SET1 CY	Set (1) CY
	CLR1 CY	Clear (0) CY
	NOT1 CY	Invert the content of CY
	SKT CY	Skip if the content of CY is 1
Bit Boolean instruction	AND1 CY, fmem.bit	Take AND/OR/XOR of the contents bit of the specified bit and CY and store the result in CY
	OR1 CY, fmem.bit	
	XOR1 CY, fmem.bit	
Interrupting process	At interrupt execution	Save to the stack memory in 8 bits parallel to the other PSW bits
	RETI	Restore from the stack memory in parallel to the other PSW

Example Take AND of bit 3 at address 3FH and P33 and set the result in CY.

```

SET1      CY          ; CY← 1
SKT       3FH. 3     ; Skip if bit 3 at address 3FH is 1
CLR1      CY          ; CY← 0
AND1      CY, PORT 3. 3 ; CY← CY∧P33
    
```

(2) Skip flag (SK2, SK1, SK0)

The skip flag is a flag to store the skip status. It is set/reset automatically as the CPU executes an instruction. It is impossible for the user to operate it directly by the program.

(3) Interrupt status flag (IST0)

The interrupt status flag is a flag to store the status of the process under execution (for details, see Table 6-3). It is impossible for the user to operate it directly by the program.

Table 4-2 Interrupt Status Flag Indication Content

IST0	Status of Process under Execution	Processing Content and Interrupt Control
0	Status 0	In a normal programming process. Enable to accept any interrupt.
1	Status 1	In an interrupting process. Disable to accept any interrupt.

The content of IST0 is saved to the stack memory as part of the PSW if the interrupt is accepted and then set automatically to 1 and set to 0 by the RETI instruction.

As it is impossible to operate IST0 by an instruction, it is always that IST0 = 1 during the interrupting process. So it is impossible to multiplex interrupts, all the interrupt requests having occurred during the interrupting process are pending until the interrupting process under execution ends (for details, see **CHAPTER 6 “INTERRUPT FUNCTIONS”**).

CHAPTER 5. PERIPHERAL HARDWARE FUNCTIONS

5.1 DIGITAL INPUT/OUTPUT PORTS

The μ PD75402A has the following digital input/output ports on chip: Ports 0 through 3, 5 and 6.

The μ PD75402A uses memory mapped I/O, and all input/output ports are mapped onto data memory space.

All data memory handling instructions can be used on all of the ports, and a wide variety of bit operations can be performed in addition to 4-bit input/output.

Note On the μ PD75402A it is not possible to perform 8-bit input/output by pairing two ports.

Fig. 5-1 Digital Input/Output Port Data Memory Addresses

Address	3	2	1	0	Symbol
FF0H	P03	P02	P01	P00	PORT0
FF1H	0	P12	0	P10	PORT1
FF2H	P23	P22	P21	P20	PORT2
FF3H	P33	P32	P31	P30	PORT3

FF5H	P53	P52	P51	P50	PORT5
FF6H	P63	P62	P61	P60	PORT6

5.1.1 Digital Input/Output Port Types, Characteristics and Configuration

The different types of digital input/output ports are shown in Table 5-1, and the configuration of each port is shown in Figs. 5-2, 5-3, 5-4 and 5-5.

Table 5-1 Digital Input/Output Port Types and Characteristics

Port (Symbol)	Function	Operation/Characteristics	Remarks
PORT0 PORT1	4-bit input	Can always be read or tested regardless of dual-function pin operating mode.	Pins also function as SO/SB0, SI, SCK, INT0, INT2. (See chapter 1 .)
PORT3*	4-bit input/output	Can be set to input or output mode bit by bit.	
PORT2 PORT6*		Can be set to input or output mode as a 4-bit unit.	Port 2 pins also function as PCL.
PORT5*	4-bit input/ output (N-ch open-drain, up to 10 V)	Can be set to input or output mode as a 4-bit unit.	Internal pull-up resistor can be specified bit by bit by mask option. (μ PD75402A only.)

* LED direct drive capability

On the μ PD75402A, a pull-up resistor can be incorporated on chip for all port pins except pins P00, P10. On the μ PD75P402, a pull-up resistor can be incorporated on chip for all port pins except pins P00, P10, and P50 through P53 (see section 5.5).

P10 has a dual function as the external vectored interrupt input pin, and has an on-chip circuit for noise elimination by the sampling clock (see section 6.2 for details).

Upon RESET input the output latches of ports 2, 3, 5 and 6 are cleared, the output buffer is turned off, and input mode is set.

Fig. 5-2 Configuration of Ports 0 and 1

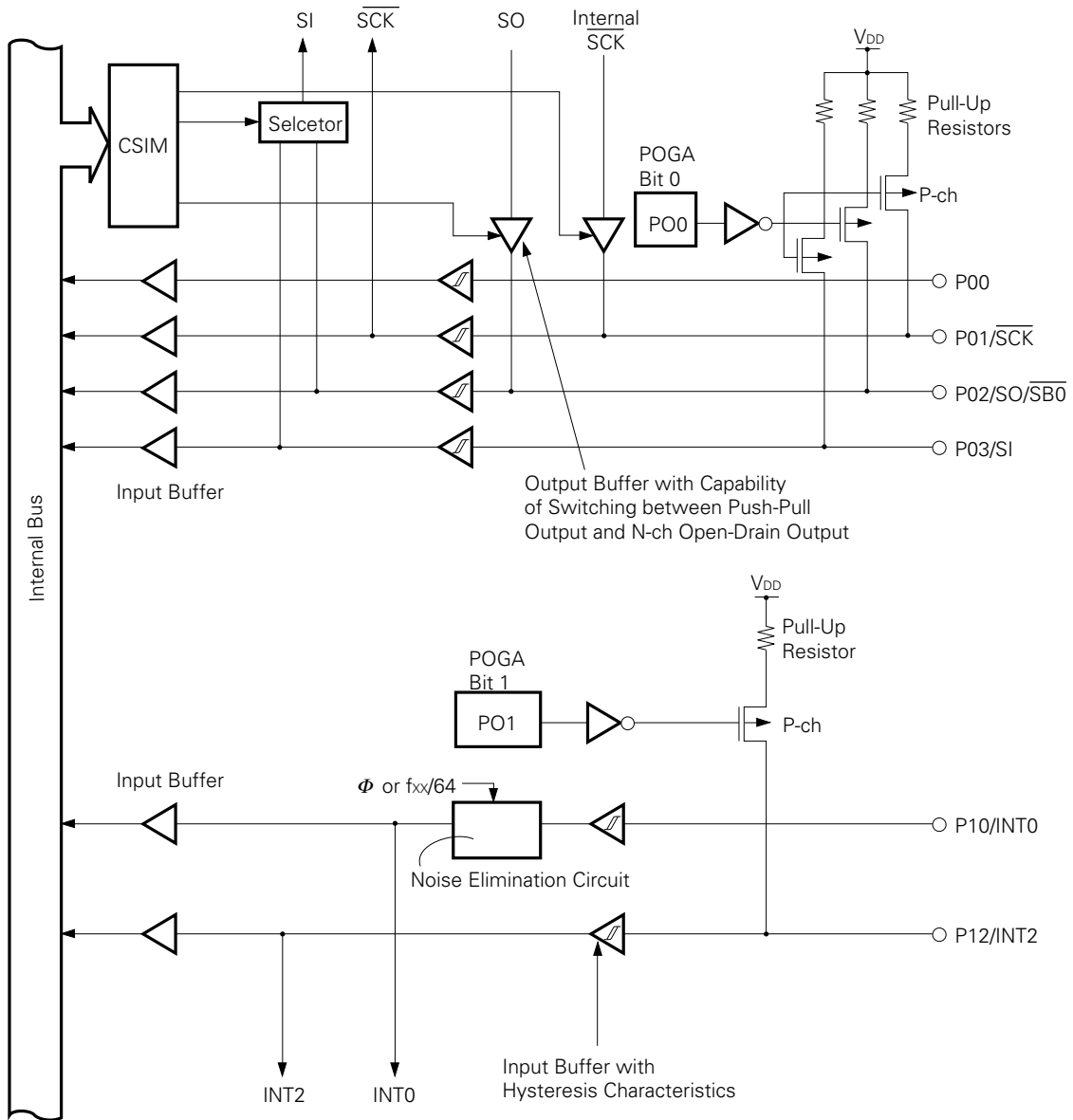
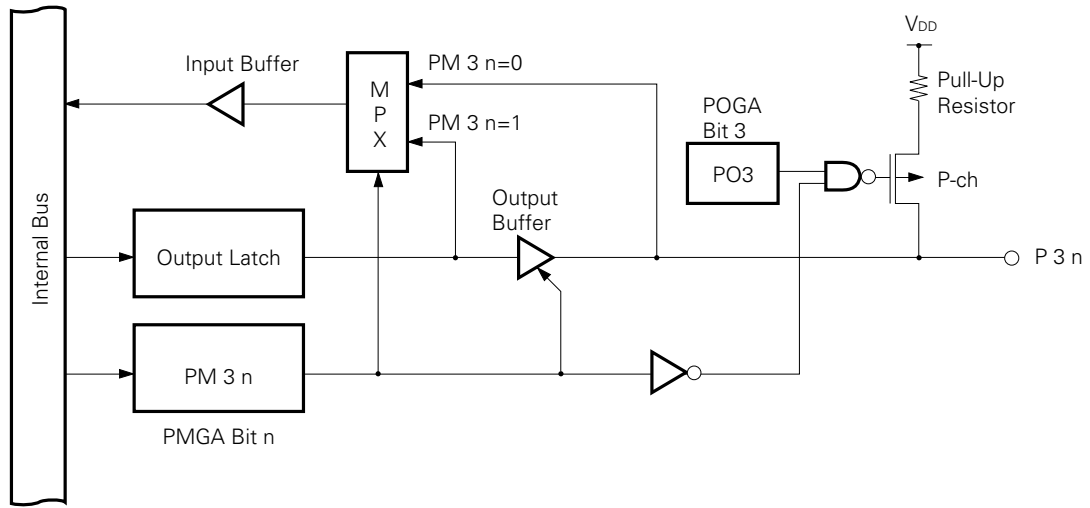
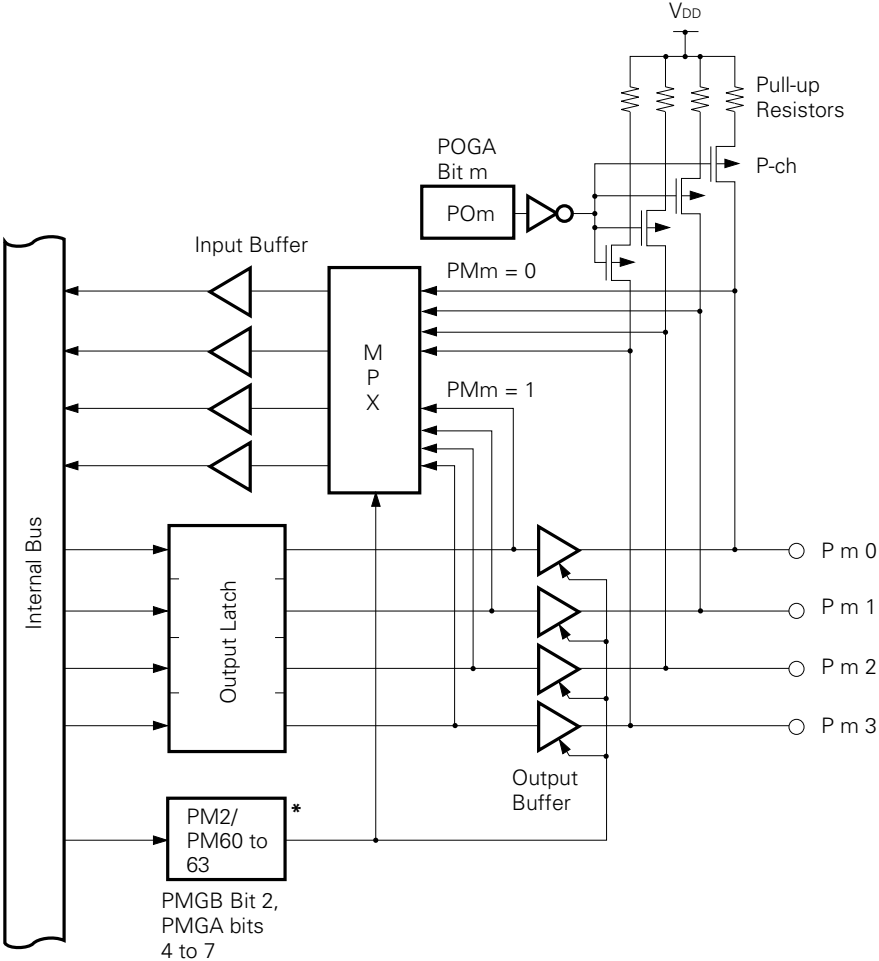


Fig. 5-3 Configuration of Port 3



Remarks n = 0 to 3

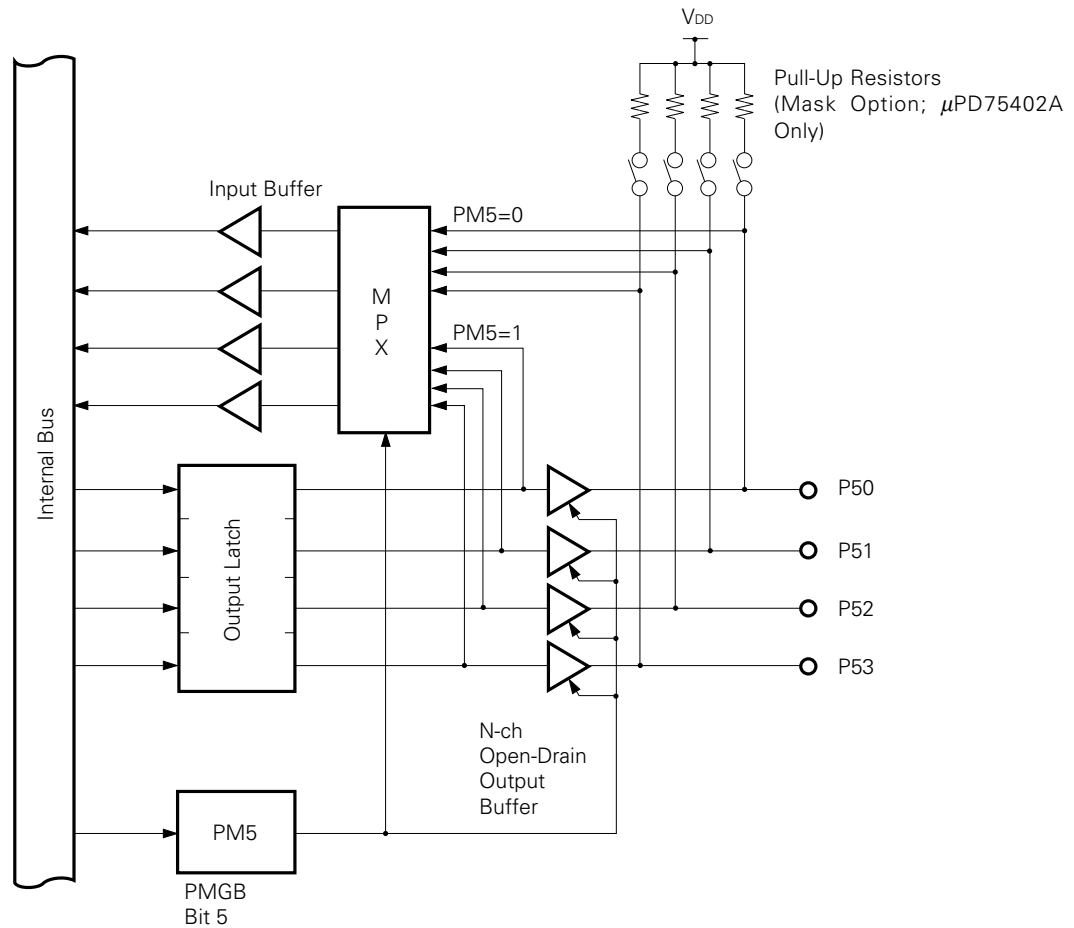
Fig. 5-4 Configuration of Ports 2 and 6



* Input/output mode specification is performed by bit 2 (PM2) of PMGB for port 2 and by bits 4 to 7 (PM60 to 63) of PMGA for port 6.

Remarks m = 2 or 6

Fig. 5-5 Configuration of Port 5



5.1.2 Input/Output Mode Setting

The input/output mode for each input/output port is set by a port mode register as shown in Fig. 5-6. For port 3, input/output can be specified bit by bit by port mode register group A (PMGA). Input/output is specified in 4-bit units by PMGB for ports 2 and 5, and by PMGA for port 6.

Each port operates as an input port when the corresponding port mode register bit is "0", and as an output port when "1".

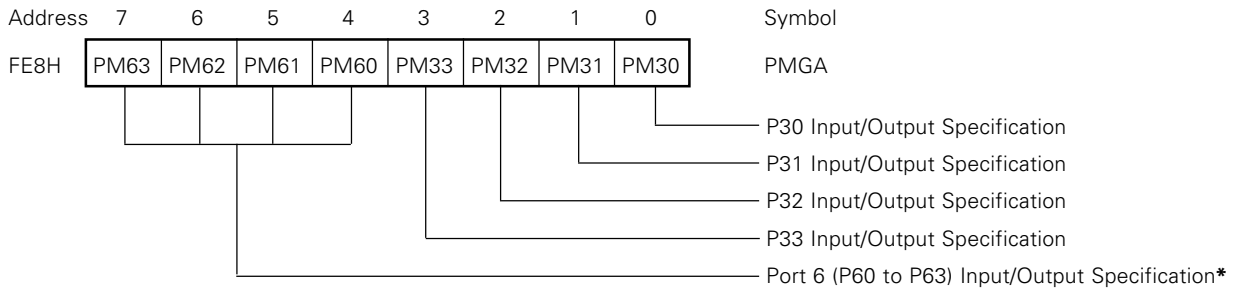
Since, when output mode is selected by setting the port mode register, the output latch contents are simultaneously output to the output pin, the output latch contents must be overwritten in advance with the required value before output mode is set.

Port mode register group A and B are each set by an 8-bit memory handling instruction.

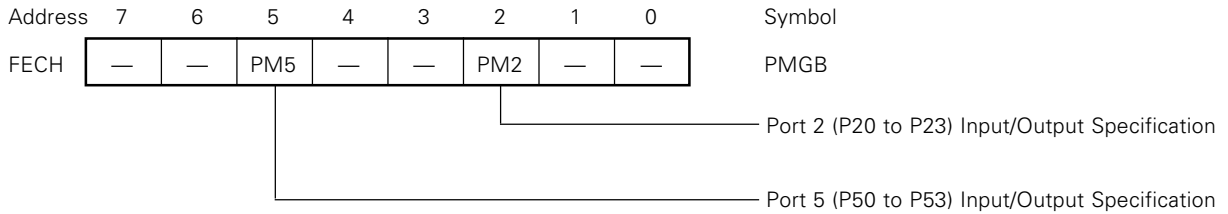
With a $\overline{\text{RESET}}$ input, all bits of each port mode register are cleared to zero, and thus the output buffer is turned off and all ports are set to input mode.

Fig. 5-6 Format of Port Mode Registers

Port Mode Register Group A



Port Mode Register Group B



	Specification
0	Input mode (output buffer off)
1	Output mode (output buffer on)

* Port 6 input/output specification is performed as a 4-bit unit. Ensure that 0000 or 1111 is written to PMGA bits 7 through 4.

5.1.3 Digital Input/Output Port Handling Instructions

As all the input/output ports in the μ PD75402A are mapped onto data memory space, all data memory handling instructions can be used. Those data memory handling instructions which are considered particularly useful for input/output pin operations are shown in Table 5-2 together with their scope of application.

(1) Bit handling instructions

Direct addressing of specific address bits (fmem.bit) can be used on all digital input/output ports.

Example To OR P50 and P31 and output the result to P61.

```

SET1    CY                ; CY ← 1
AND1    CY, PORT5.0      ; CY ← CY ∧ P50
OR1     CY, PORT3.1      ; CY ← CY ∨ P31
SKT     CY
BR      CLRP
SET1    PORT6.1          ; P61 ← 1
      ⋮
      ⋮
CLRP : CLR1    PORT6.1    ; P61 ← 0
    
```

(2) 4-bit handling instructions

All 4-bit memory handling instructions can be used, such as MOV, XCH, ADDS and INCS, in addition to the IN/OUT instructions.

Example 1. To output accumulator contents to port 3.

```
OUT    PORT3, A
```

2. To add the accumulator value to the data output to port 5, and output the result.

```

MOV    HL, #PORT5
ADDS   A, @HL      ; A ← A + PORT5
NOP
MOV    @HL, A      ; PORT5 ← A
    
```

Table 5-2 List of Input/Output Pin Handling Instructions

	PORT 0	PORT 1	PORT2	PORT 3	PORT5	PORT 6
IN A, PORTn	○					
OUT PORTn, A	-	-	○			
SET1 PORTn.bit	-	-	○			
CLR1 PORTn.bit	-	-	○			
SKT PORTn.bit	○					
SKF PORTn.bit	○					
AND1 CY, PORTn.bit	○					
OR1 CY, PORTn.bit	○					
XOR1 CY, PORTn.bit	○					

5.1.4 Digital Input/Output Port Operations

Port and port pin operations when a data memory handling instruction is executed for a digital input/output port differ according to the input/output mode setting (see Table 5-3). This is because, as can be seen from the input/output port configurations, data fetched onto the internal bus is treated as pin data in input mode and as output latch data in output mode.

(1) Operations when input mode is set

When a test instruction such as the SKT instruction, or an instruction which fetches port data as 4 bits onto the internal bus (IN, MOV and bit operation instructions) is executed, individual pin data is manipulated.

When an instruction which performs a 4-bit transfer of accumulator contents to a port (OUT or MOV instruction) is executed, the accumulator data is latched in the output latch. The output buffer remains off.

When an XCH instruction is executed, the individual pin data is input to the accumulator, and the accumulator data is latched in the output latch. The output buffer remains off.

When an INCS instruction is executed, data comprising the individual pin data (4-bit) + 1 is latched in the output latch. The output buffer remains off.

When a bit-wise data memory rewriting instruction such as the SET1/CLR1/SKTCLR instructions is executed, the output latch for the specified bit can be rewritten as directed by the instruction, but the contents of the output latches for the other bits are undefined.

(2) Operations when output mode is set

When a test instruction, bit input instruction, or an instruction which fetches port data as 4 bits onto the internal bus is executed, the output latch contents are manipulated.

When an instruction which performs a 4-bit transfer of accumulator contents is executed, when the output latch data is overwritten it is simultaneously output from the pins.

When an XCH instruction is executed, the output latch contents are transferred to the accumulator, and the accumulator contents are latched in the output latches and output from the pins.

When an INCS instruction is executed, data comprising the output latch contents + 1 is latched in the output latches and output from the pins.

When a bit output instruction is executed, the specified output latch bit is rewritten and output from the pin.

Table 5-3 Operations with Input/Output Port Handling Instructions

Instruction Executed	Port and Individual Pin Operations	
	Input Mode	Output Mode
SKT PORTn.bit SKF PORTn.bit	Tests pin data.	Tests output latch data.
AND1 CY, PORTn.bit OR1 CY, PORTn.bit XOR1 CY, PORTn.bit	Operation between pin data and CY	Operation between output latch data and CY
IN A, PORTn MOV A, PORTn	Transfers pin data to accumulator.	Transfers output latch data to accumulator.
OUT PORTn, A MOV PORTn, A	Transfers accumulator data to output latches. (Output buffer remains off.)	Transfers accumulator data to output data to output latches, and outputs data from pins.
XCH A, PORTn	Transfers pin data to accumulator, and transfers accumulator data to output latches. (Output buffer remains off.)	Exchanges data between output latches and accumulator.
INCS PORTn	Latches pin data + 1 in output latches.	Increments output latch contents by 1.
SET1 PORTn.bit CLR1 PORTn.bit SKTCLR PORTn.bit	Specified bit output latch is rewritten as per instruction, but output latches for other bits are undefined.	Changes output pin status as per instruction.

5.1.5 Internal Pull-up Resistors

The μ PD75402A can incorporate internal pull-up resistors for all port pins except P00 and P10. The μ PD75P402 can incorporate internal pull-up resistors for all port pins except P00, P10, and P50 through P53.

As shown in Table 5-4, internal pull-up resistors can be specified by software or by a mask option (although specification by mask option is not possible on the μ PD75P402).

The pull-up resistor specification register which specifies internal pull-up resistors for ports 0 through 3 and 6 (POGA) is an 8-bit write-only register. Its format is shown in Fig. 5-7. POGA is set by an 8-bit write instruction. Reading and bit manipulation is not possible. RESET input clears this register to zero.

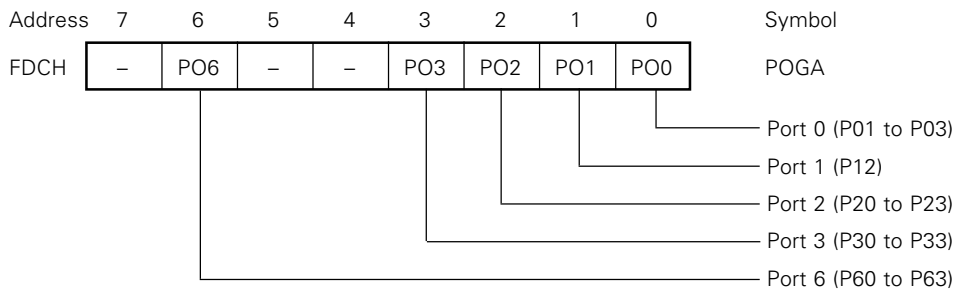
Internal pull-up resistor specification for port 3 is valid only for input pins specified as in input mode. The status of pins specified as in output mode is “without pull-up resistor” irrespective of the POGA setting.

Table 5-4 Internal Pull-Up Resistor Specification for Each Port

Port (Pin Name)	Internal Pull-Up Resistor Specification Method	Specification Bits
Port 0 (P01 to P03)*1	3-bit unit internal specification by software	POGA.0
Port 1 (P12)*1	Internal specification by software	POGA.1
Port 2 (P20 to P23) Port 3 (P30 to P33) Port 6 (P60 to P63)	4-bit unit internal specification by software	POGA.2 POGA.3 POGA.6
Port 5 (P50 to P53)	Bit-wise internal specification by mask option*2	—

- * 1. Pull-up resistors cannot be incorporated into the P00 and P10 pins.
- 2. On the μ PD75P402, pull-up resistors cannot be incorporated into port 5.

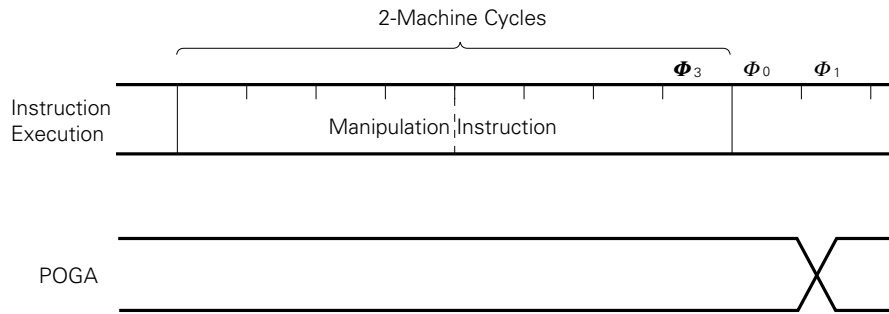
Fig. 5-7 Format of Pull-Up Resistor Specification Register



	Specification
0	Pull-up resistor not incorporated
1	Pull-up resistor incorporated

The timing for switching of pull-up resistor presence/ absence by the setting of the pull-up resistor specification register (POGA) is shown in Fig. 5-8.

Fig. 5-8 Pull-Up Resistor Incorporation Switching Timing



Remarks Φ_0 through Φ_3 are internal operation timing clock pulses.

After pull-up resistor incorporation has been specified by overwriting POGA, in consideration of the external load capacity the pin level should be stabilized by execution of an NOP instruction etc. before an input/output instruction is executed.

Example To perform input after specifying incorporation of a pull-up resistor in port 1.

```

MOV     XA, #02H           ; Pull-up resistor incorporated in port 1
MOV     POGA, XA
      ⋮
      ⋮
IN      A, PORT1
    
```

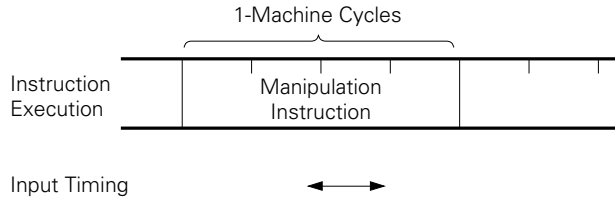
} Wait until pin level is stabilized in consideration of external load capacity.

5.1.6 Digital Input/Output Port Input/Output Timing

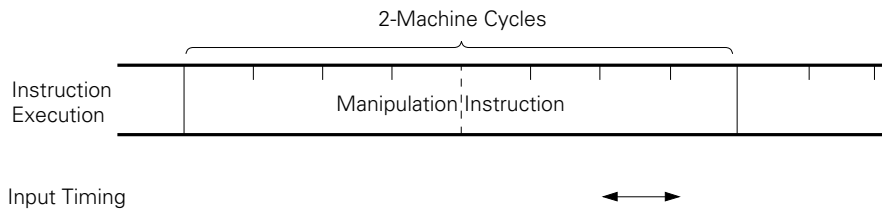
The timing for outputting data to the output latch and fetching pin data or output latch data onto the internal bus is shown in Fig. 5-9.

Fig. 5-9 Digital Input/Output Port Input/Output Timing

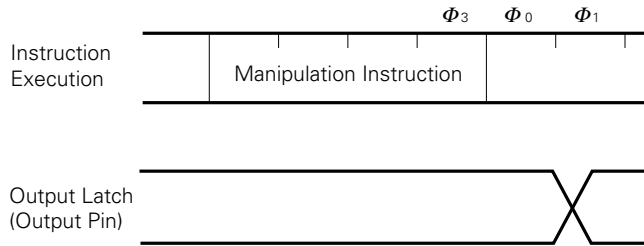
(a) Data fetch by 1-machine-cycle instruction



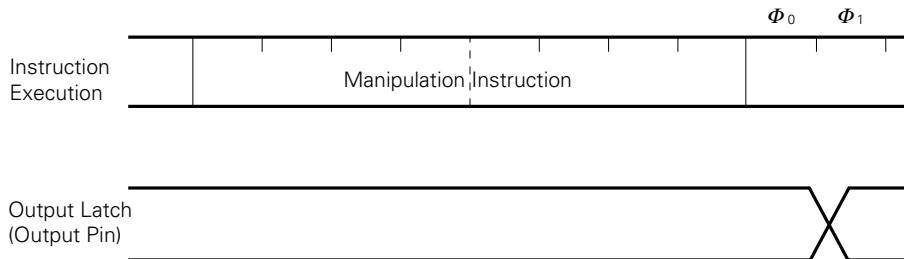
(b) Data fetch by 2-machine-cycle instruction



(c) Data latching by 1-machine-cycle instruction



(d) Data latching by 2-machine-cycle instruction



5.2 CLOCK GENERATION CIRCUIT

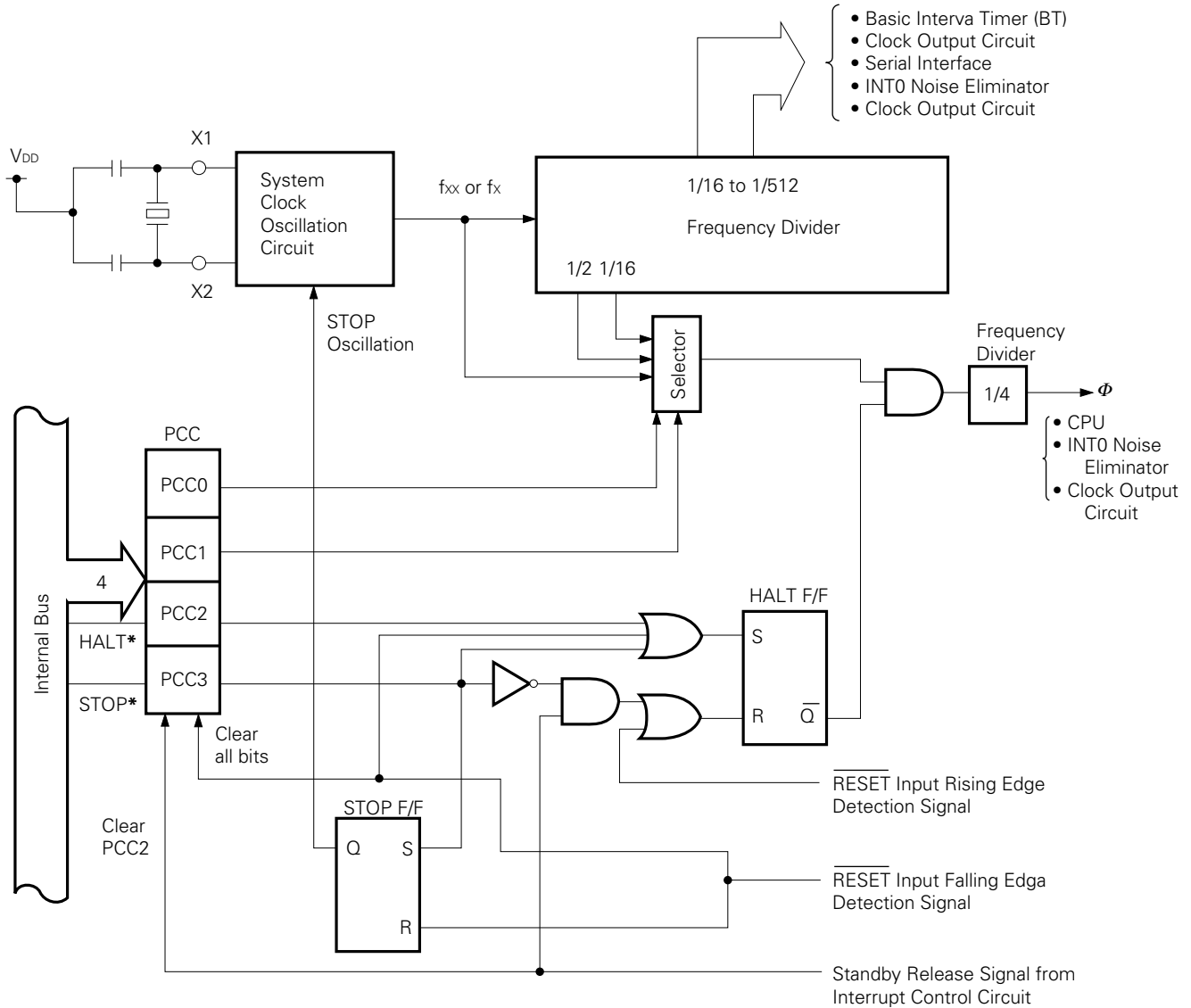
The clock generation circuit supplies various clocks to the CPU and peripheral hardware, and controls the operating mode of the CPU.

5.2.1 Clock Generation Circuit Configuration

The configuration of the clock generation circuit is shown in Fig. 5-10.

★

Fig. 5-10 Clock Generation Circuit Block Diagram



- Remarks**
1. f_{xx} = System clock frequency
 2. f_x = External clock frequency
 3. PCC: Processor clock control register
 4. 1 clock cycle (fcy) of Φ is 1 machine cycle of an instruction.

* Instruction execution

5.2.2 Clock Generation Circuit Function and Operation

The clock generation circuit generates the CPU clock (Φ) and various clocks for supply to peripheral hardware, and controls the CPU operating mode, such as standby mode etc.

Clock generation circuit operation is determined by the processor clock control register (PCC).

Upon RESET input, the PCC is cleared to 0000 and the μ PD75402A operates in low-speed mode (15.3 μ s: when operating at 4.19 MHz).

Peripheral hardware is supplied with various clocks scaled from the system clock generation circuit output (f_{xx} in the case of crystal/ceramic oscillation, and f_x when an external clock is used) by the frequency divider.

From this section on, only f_{xx} is used when expressing the speed of the various clocks; for an external clock, this should be replaced by f_x .

The operation of each block is described below.

(1) Processor clock control register (PCC)

The PCC is a 4-bit register which performs selection of the CPU clock Φ and control of the CPU operating mode.

The format of the PCC is shown in Fig. 5-11.

When bit 3 or bit 2 is set (1), standby mode is selected. When standby mode is released by the Standby Release signal, both bits are automatically cleared and the normal operating mode is reestablished (see **CHAPTER 7 "STANDBY FUNCTION"** for details). Setting of the low-order 2 bits of the PCC is performed by a 4-bit memory handling instruction. At this time, ensure that bits 3 and 2 are reset to "0" so that the pattern "00xx" is written.

Bits 3 and 2 are set (1) by the STOP instruction and the HALT instruction respectively.

RESET input clears the PCC to 0000.

Example 1. To set the machine cycle to 0.95 μ s ($f_{xx} = 4.19$ MHz).

```
SEL      MB15
MOV      A, #0011B
MOV      PCC, A
```

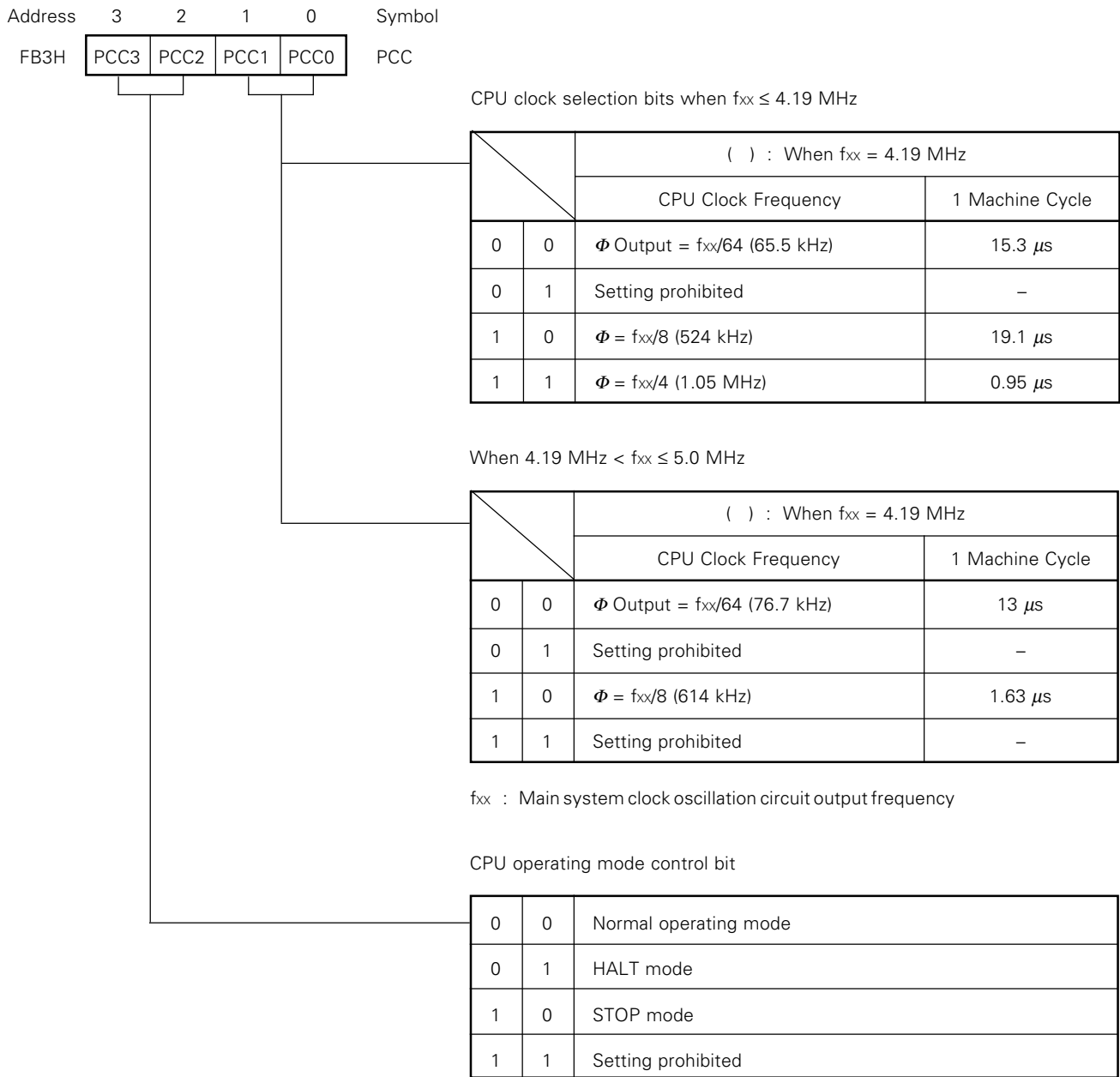
2. To set the machine cycle to 1.63 μ s ($f_{xx} = 4.91$ MHz).

```
SEL      MB15
MOV      A, # 0010B
MOV      PCC, A
```

3. To select the STOP mode. (Be sure to include an NOP instruction after a STOP or HALT instruction.)

```
STOP
NOP
```

Fig. 5-11 Processor Clock Control Register Format



Note When using a value of f_{xx} such that 4.19 MHz < $f_{xx} \leq 5.0$ MHz, if maximum speed mode: $\Phi f_{xx}/4$ (PCC1, PCC0 = 11) is set as CPU clock frequency, 1 machine cycle is less than $0.95 \mu s$ and the standard minimum value $0.95 \mu s$ is not kept.

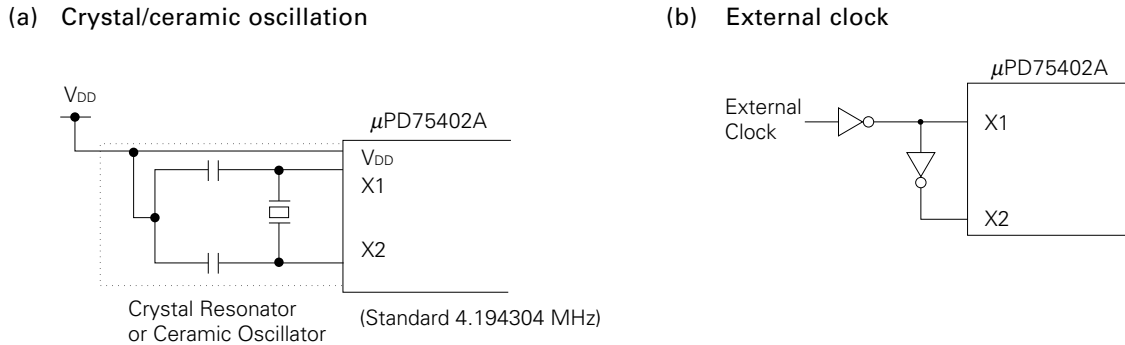
Therefore, in this case, PCC1, PCC0 = 11 cannot be set, so it should be used with PCC1, PCC0 = 00 or 10. As a result, the combination of " $f_{xx} = 4.19$ MHz, PCC1, PCC0 = 11" is maximum speed selection (1 machine cycle = $0.95 \mu s$) as CPU clock.

(2) System clock oscillation circuit

The system clock oscillation circuit oscillates by means of a crystal resonator or ceramic resonator connected to the X1 and X2 pins (standard: 4.194304 MHz).

An external clock can also be input.

Fig. 5-12 System Clock Oscillation Circuit External Circuitry



- Note**
1. When an external clock is input, STOP mode cannot be set. If STOP mode is set, the X1 pin is shorted to V_{SS} (GND potential) internally to suppress clock oscillation circuit leakage.
 2. When using the system clock generation circuit the area enclosed in dotted line in Fig. 5-12 should be wired in order to avoid effects of wiring capacitance etc., as shown below.

- Minimize the length of wiring
- Do not cross other signal lines, or position wiring close to a variable high current.
- The connecting point of the oscillator capacitor should always be of the same potential as V_{DD}. Do not connect it to the supply pattern where there is a high current.
- Do not pick up the signal from the oscillator.

Fig. 5-13 Example of Poor Resonator Connection Circuit (1/2)

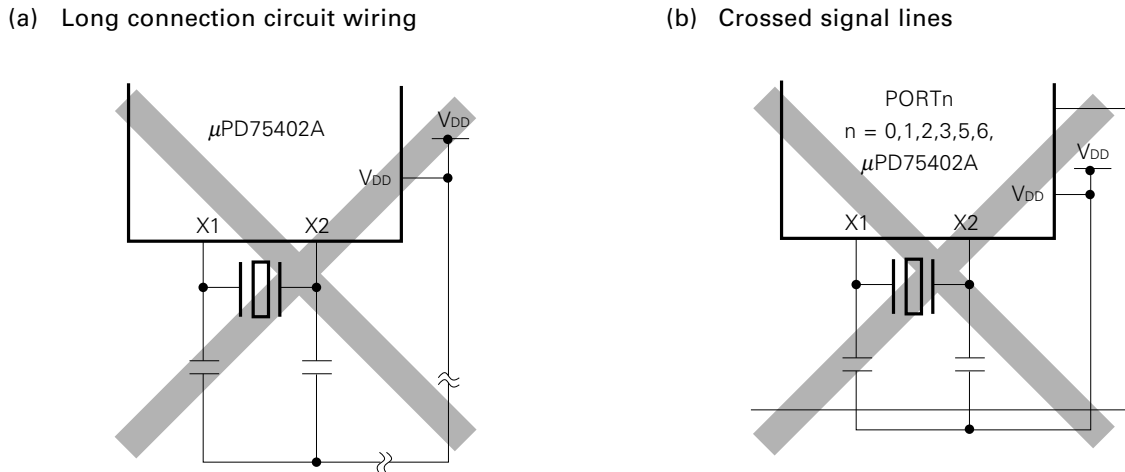
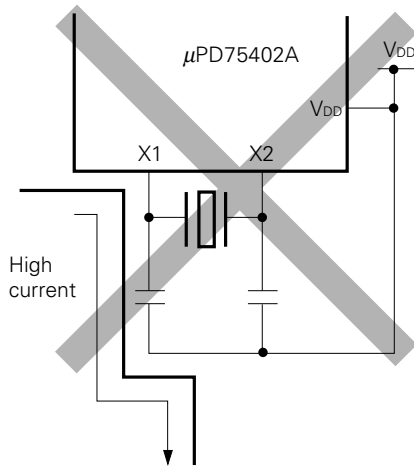
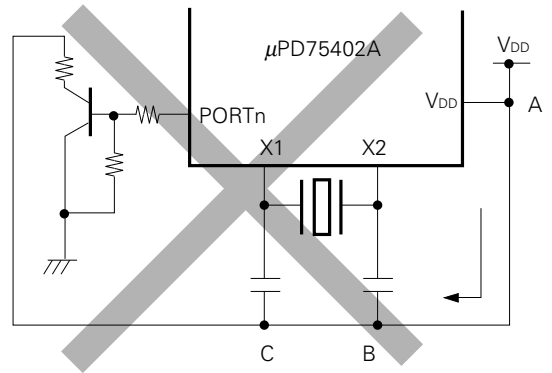


Fig. 5-13 Example of Poor Resonator Connection Circuit (2/2)

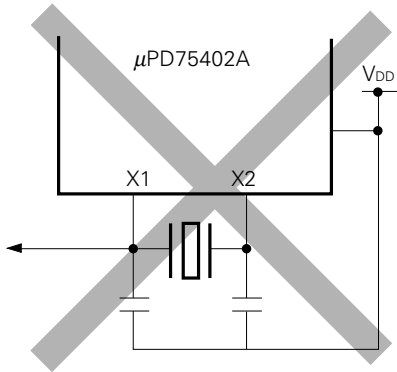
(c) Signal line close to varyin high current



(d) Current flows an oscillator power supply line.
(potentials at A, B and C fluctuate.)



(e) Signal is picked up.

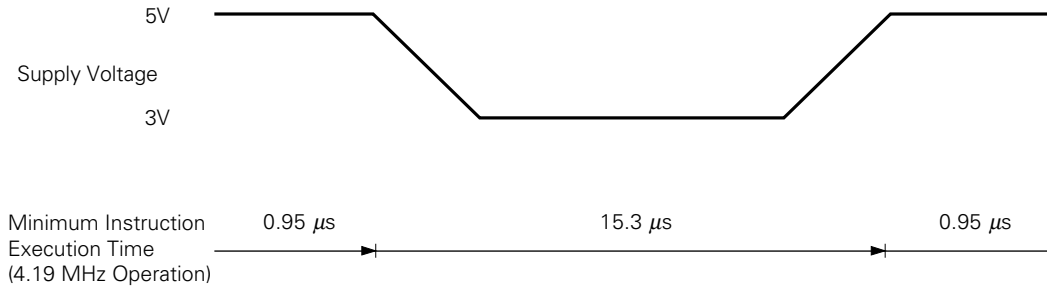


5.2.3 CPU Clock Setting

The CPU clock Φ is the clock supplied to the μ PD75402A's internal CPU, and the reciprocal of this clock is the minimum instruction execution time (defined in this manual as 1 machine cycle).

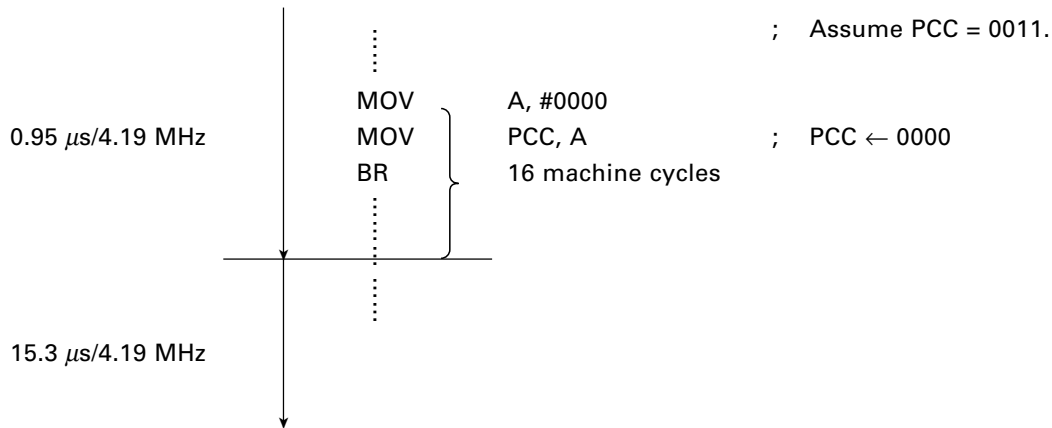
On the μ PD75402A, Φ can be switched in 3 steps by setting the PCC. In other words, with the same system clock oscillator frequency f_{xx} , the minimum instruction execution time can be changed in 3 steps. By using this function to select operation at high speed ($0.95 \mu\text{s}$: When operating at 4.19 MHz) when the power supply voltage is 5 V and low-speed operation ($15.3 \mu\text{s}$: When operating at 4.19 MHz) in backup mode, applications can be implemented in which reduced current consumption and low-power operation is possible, which is an extremely useful facility.

Fig. 5-14 Use of Variable Minimum Instruction Execution Time Function



Note As explained in the previous section, Φ is changed by a 4-bit memory handling instruction, but after a PCC change, operation is at the pre-change Φ rate for the number of machine cycles shown in Table 5-5.

Example



As the PCC is set in 0 by $\overline{\text{RESET}}$ input, Φ is reset-started at the slowest speed (state in which the operating voltage range is wide). For this reason, in a system with a slow supply voltage rise (such as a system with a high- capacitance capacitor connected), correct operation is possible even when an adequate supply voltage cannot be attained after a power-on reset.

Fig. 5-15 Change of Φ after Power-On Reset

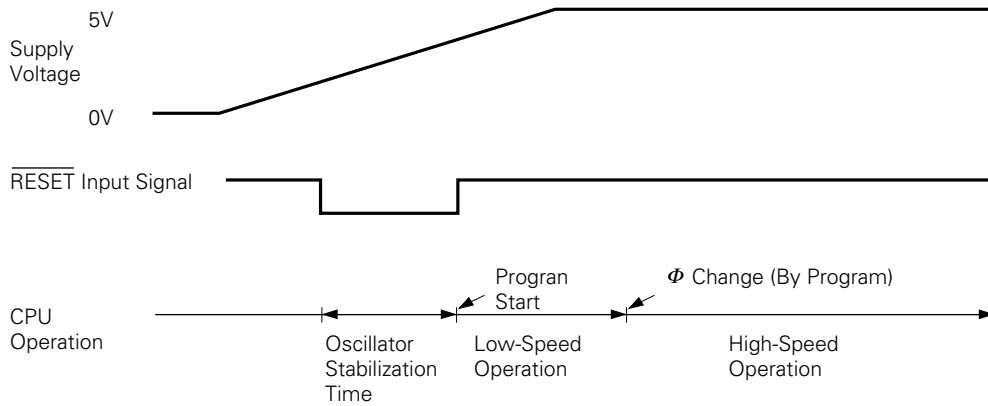


Table 5-5 Maximum Time Required for Change of CPU Clock

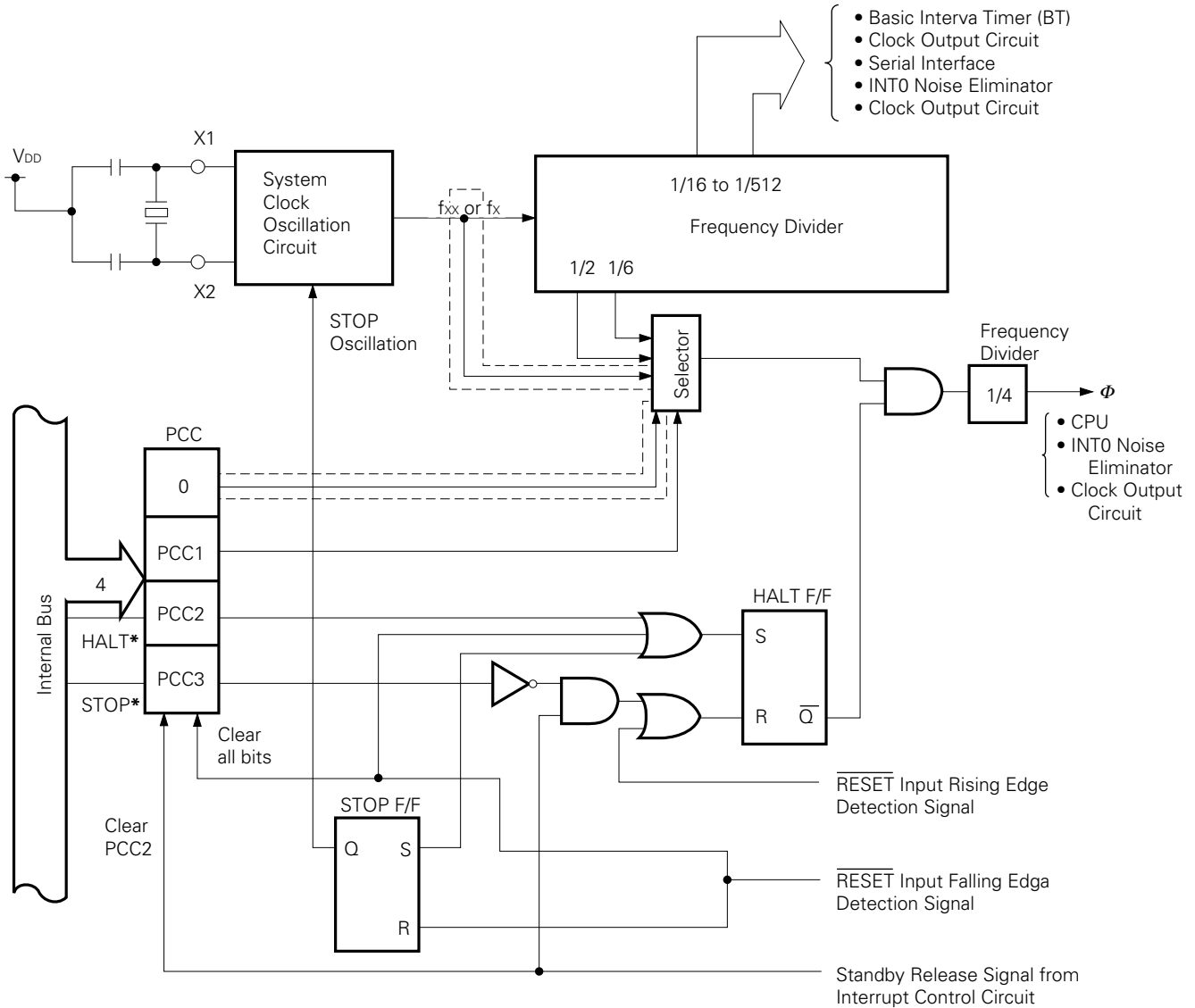
PCC Before Change	PCC After Change	Max. No. of Machine Cycles Required for Change of Φ	Max. Time Required for Change of Φ * (When fxx = 4.19 MHz)
0000	0010	1	15.3 μs
	0011	1	
0010	0000	8	
	0011	8	
0011	0000	16	
	0010	16	

* When standby mode is not set until Φ changes.

5.2.4 Differences Between μ PD75402A and μ PD75402

Part of the clock generation circuit differs between the μ PD75402A and the μ PD75402. The μ PD75402 does not include the sections enclosed in dotted lines.

Fig. 5-16 Clock Generation Circuit - Differences between μ PD75402A and μ PD75402

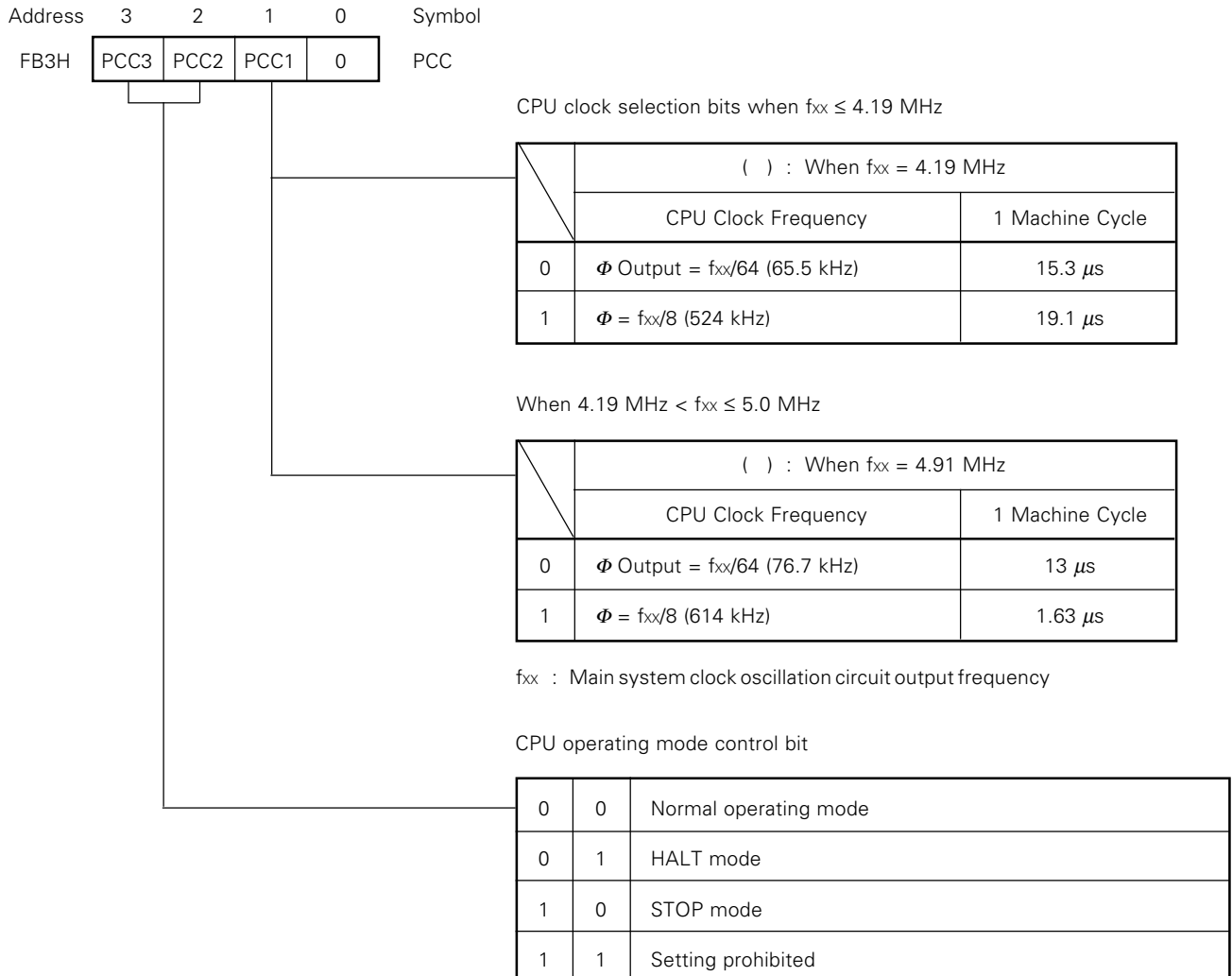


* Instruction execution

- Remarks**
1. f_{xx} = System clock frequency
 2. f_x = external clock frequency
 3. PCC: Processor clock control register
 4. 1 clock cycle (t_{cy}) of Φ is 1 machine cycle of an instruction.

Next, the processor clock control register (PCC) of the μ PD75402 is shown below. Setting of bit 1 of the PCC is performed by a 4-bit memory handling instruction. At this time, ensure that bits 3, 2 and 0 are reset to "0" so that the pattern "00 x 0" is written.

Fig. 5-17 μ PD75402 Processor Clock Control Register Format



- Note**
1. Ensure that 0 is always written to PCC bit 0.
 2. Unlike the μ PD75402A, in the μ PD75402, switching Φ is 2-step rather than 3-step. High-speed mode (0.95 μ s at 4.19 MHz) cannot be specified.

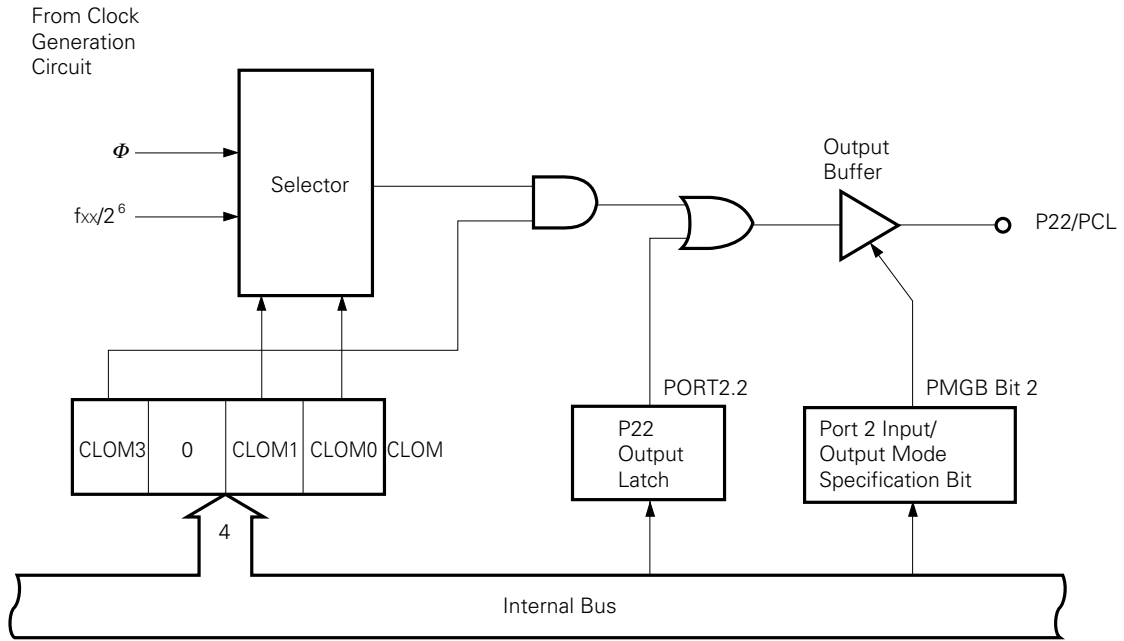
5.3 CLOCK OUTPUT CIRCUIT

The clock output circuit outputs clock pulses from the P22/PCL pin, and is used to supply clock pulses to peripheral LSIs, etc.

5.3.1 Clock Output Circuit Configuration

The configuration of the clock output circuit is shown in Fig. 5-18.

Fig. 5-18 Clock Output Circuit Configuration



Remarks When switching between clock output enabled/disabled states, consideration has been given to ensuring that a short pulse is not output.

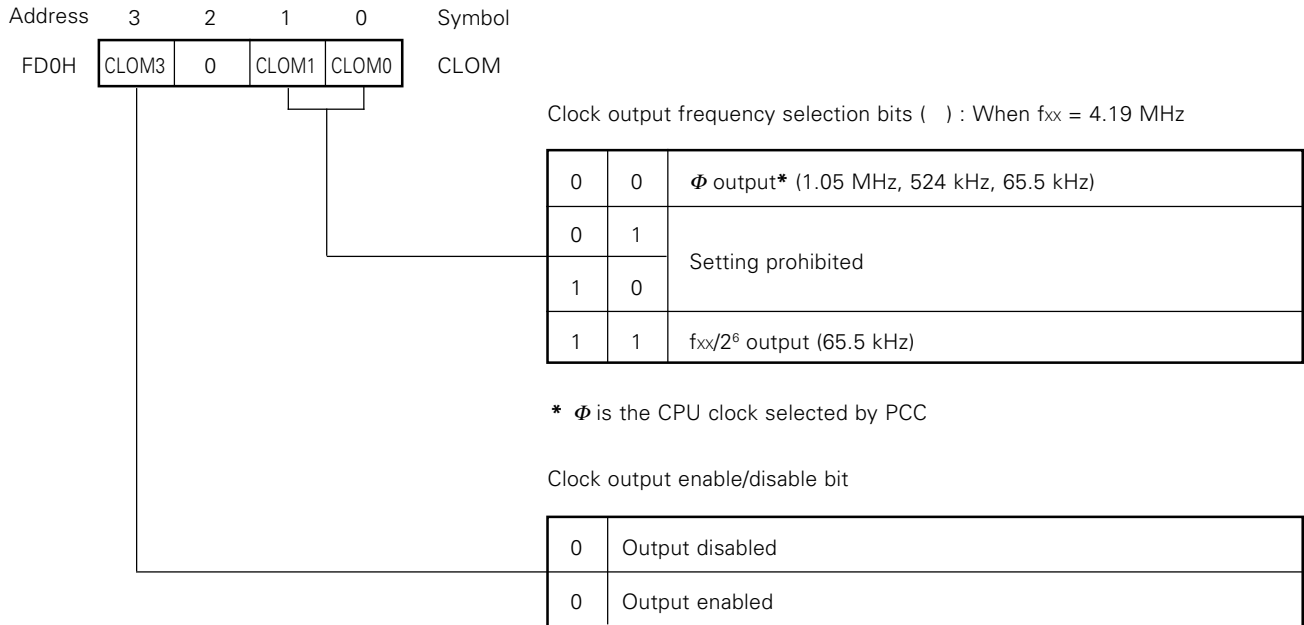
5.3.2 Clock Output Mode Register (CLOM)

CLOM is a 4-bit register used to control clock output.

CLOM is set by a 4-bit memory handling instruction. Bit handling instructions cannot be used. Also, this register cannot be read.

RESET input clears CLOM to zero and selects the clock output disabled state.

Fig. 5-19 Clock Output Mode Register Format



Note Ensure that 0 is written to bit 2 of CLOM.

5.3.3 Clock Output Procedure

Clock pulse output is performed by the following procedure.

- (i) Set the clock output mode register.
- (ii) Write 0 to the P22 output latch.
- (iii) Set the port 2 input/output mode to output.

This procedure may be reversed depending on the treatment of P22/PCL prior to clock output.

Example 1. To output a 65.5 kHz (at 4.19 MHz operation) clock from the PCL/P22 pin. (The PCL/P22 pin outputs the clock from the high-impedance state.)

```

MOV     A, #1011B
MOV     CLOM, A           ; CLOM = 1011B
CLR1    PORT2.2          ; P22 ← 0
MOV     XA, #04H
MOV     PMGB, XA         ; PMGB = 00000100B
    
```

2. To output Φ . (The PCL/P22 pin outputs the clock from the low-impedance state.)

```

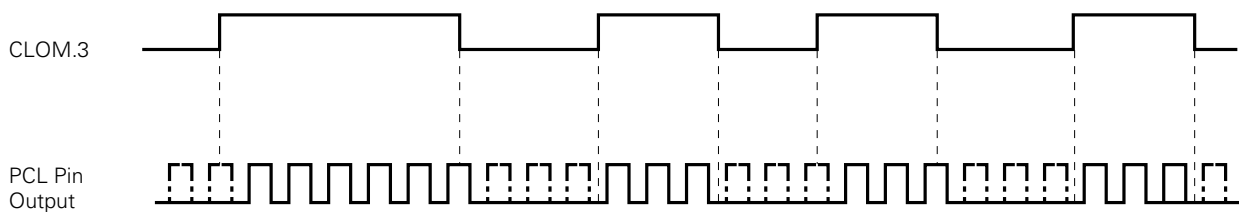
MOV     A, #0
OUT     PORT2, A         ; P22 ← 0
MOV     XA, #04H
MOV     PMGB, XA
MOV     A, #1000B
MOV     CLOM, A         ; CLOM ← 1000B
    
```

5.3.4 Example of Remote Control Application

The μ PD75402A clock output functions can be used in remote control applications. The remote control output carrier frequency is selected by the clock frequency selection bits of the clock output mode register. Pulse output enabling/disabling is performed by software control of the enable/disable bit.

When switching between clock output enabled/disabled states, consideration has been given to ensuring that a short pulse is not output.

Fig. 5-20 Example of Remote Control Application



5.4 BASIC INTERVAL TIMER

The μ PD75402A is equipped with an 8-bit basic interval timer which has the following functions:

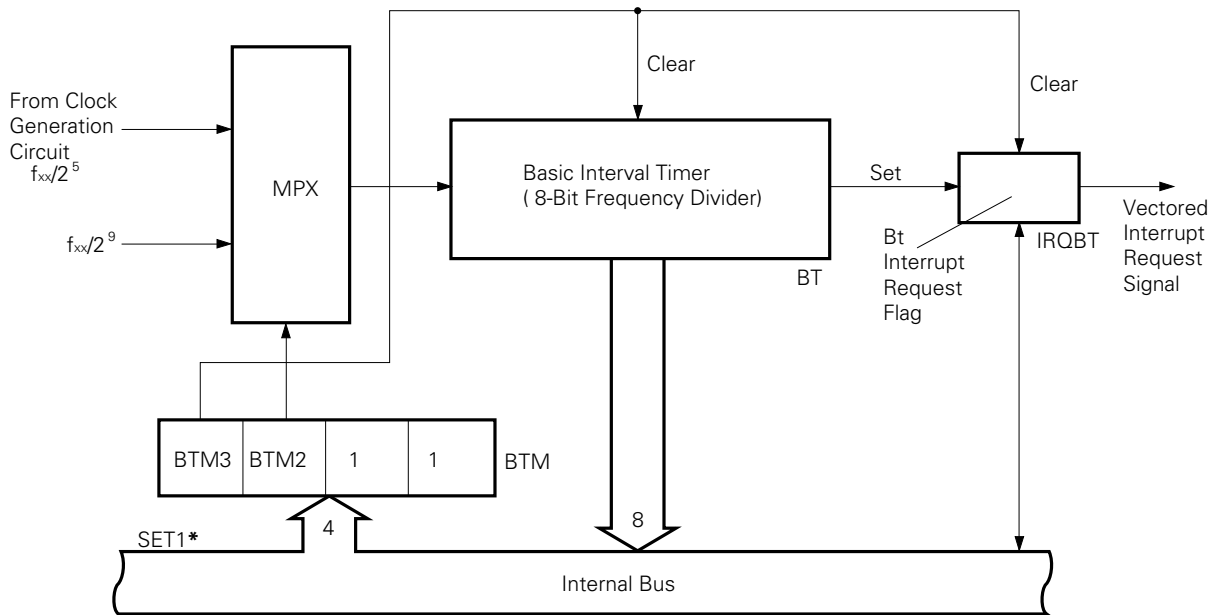
- (a) Standard time generation (2 different time intervals)
- (b) Reading counter contents

This basic interval timer can also be used as a watchdog timer for the detection of inadvertent program looping.

5.4.1 Basic Interval Timer Configuration

The configuration of the basic interval timer is shown in Fig. 5-21.

Fig. 5-21 Basic Interval Timer Configuration



* Instruction execution

5.4.2 Basic Interval Timer Mode Register (BTM)

BTM is a 4-bit register which controls the operation of the basic interval timer.

BTM is set by a 4-bit memory handling instruction. Bit operations are not possible.

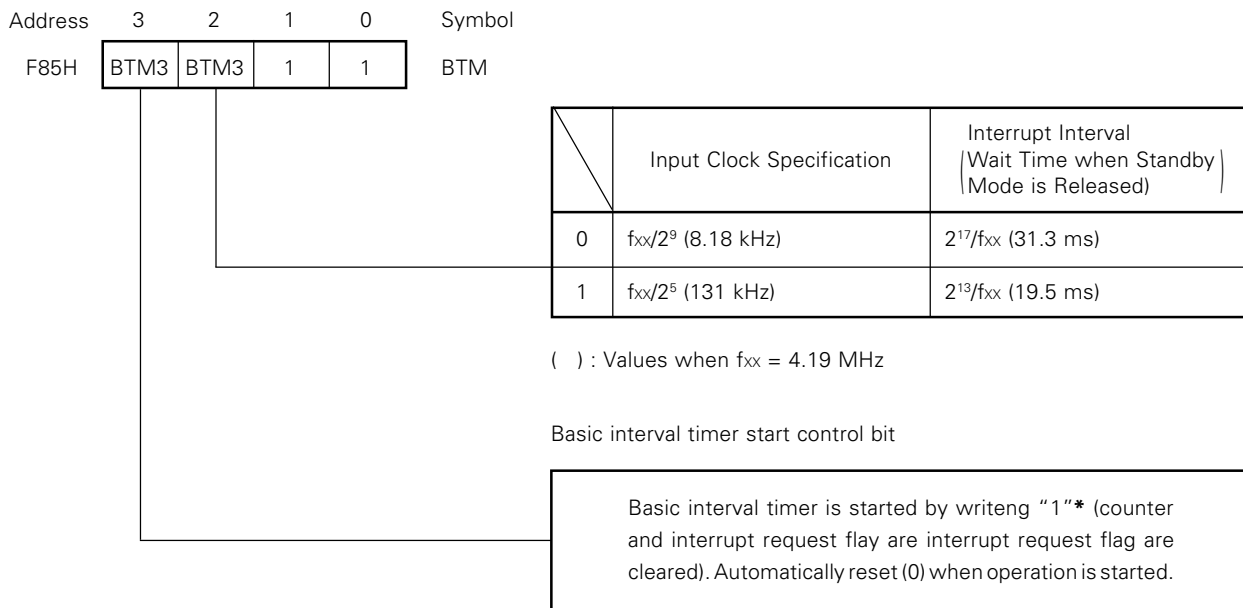
Example To set the interrupt generation interval to 1.95 ms (4.19 MHz).

```
MOV     A, #1111B
MOV     BTM. A           ; BTM ← 1111B
```

When bit 3 is set (1), the contents of the basic interval timer are cleared and at the same time the basic interval timer interrupt request flag (IRQBT) is also cleared (start of basic interval timer).

With a RESET input, BTM contents are cleared to zero and the interrupt request signal generation interval is set to the maximum length.

Fig. 5-22 Basic Interval Timer Mode Register Format



* Ensure that the write is performed by a 4-bit write instruction.

Note Ensure that 1 is written to bits 1 and 0.

5.4.3 Basic Interval Timer Operation

The basic interval timer (BT) is constantly incremented by the clock from the clock generation circuit, and sets the interrupt request flag (IRQBT) when it overflows. The BT count operation cannot be stopped.

Either of two times can be selected as the interrupt generation interval by setting the BTM (Fig. 5-22).

The basic interval timer and the interrupt request flag can be cleared by setting (1) bit 3 of BTM (directive to start as interval timer).

The basic interval timer (BT) count status can be read by an 8-bit handling instruction. Data cannot be written to the timer.

Note When the basic interval timer contents are read, it may happen that unstable data in the process of count updating is read. To prevent this, the read instruction should be executed twice, then the two read contents should be compared. If a comparison of the two read contents shows appropriate values, the latter contents are taken as the result of the read. If the values are completely different, the operation should be repeated from the beginning.

Example To read the BT count contents

```

MOV      HL, #TEMP      ; Set table address to HL
LOOP:   MOV      XA, BT   ; First read
MOV      TEMP, XA      ; Save read value
MOV      XA, BT        ; Second read
SKE      A, @HL        ; Comparison of low-order 4 bits
BR       LOOP
INCS     L
XCH      A, X
SKE      A, @HL        ; Comparison of high- order 4 bits
BR       LOOP

```

5.4.4 Examples of Basic Interval Timer Applications

Example 1. In this example the basic interval timer is enabled, and the interrupt generation interval is set to 1.95 ms (at 4.19 MHz operation).

```

SEL      MB15
MOV      A, #1111B
MOV      BTM,A          ; Setting and start
EI       ; Enable interrupts
EI       IEBT           ; Enable BT interrupts
    
```

Example 2. Example of watchdog timer application

When used as a watchdog timer, the basic interval timer’s function of generating an interrupt (INTBT) at set intervals is used.

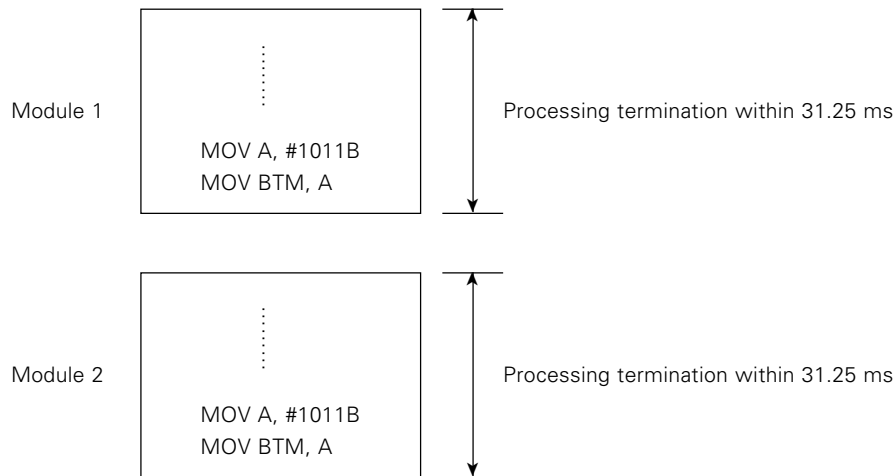
First, the standard time generated by the basic interval timer is decided.

Next, the program is divided into a number of modules whose processing should terminate within the standard time, and the counter (BT) and interrupt request flag (IRQBT) are cleared each time a module ends. A program is written such that an interrupt (INTBT) is not generated if operation is normal. In other words, if an interrupt is generated, this is interpreted as indicating inadvertent program looping.

The interrupt generation interval is set to 31.25 ms (in 4.19 MHz operation) so that the processing time for each module is less than 31.25 ms.

```

Initial Setting {
SEL      MB15          ; or CLR1 MBE
MOV      A, #1011B
MOV      BTM, A       ; Interval setting and counter start
EI
EI       IEBT
    
```



5.5 SERIAL INTERFACE

5.5.1 Serial Interface Functions

The μ PD75402A incorporates a clocked 8-bit serial interface, with the following three modes available.

(1) Operation-halted mode

This mode is used when no serial transfer is to be performed, and allows power dissipation to be reduced.

(2) 3-wire serial I/O mode

In this mode, 8-bit data transfer is performed using three lines: The serial clock (\overline{SCK}), serial output (SO), and serial input (SI).

In the 3-wire serial I/O mode simultaneous transmission and reception is possible, increasing the data transfer processing speed.

The 3-wire serial I/O mode allows connection to 75X series and 78K series devices and various peripheral I/O devices.

Serial data transfer is performed MSB-first.

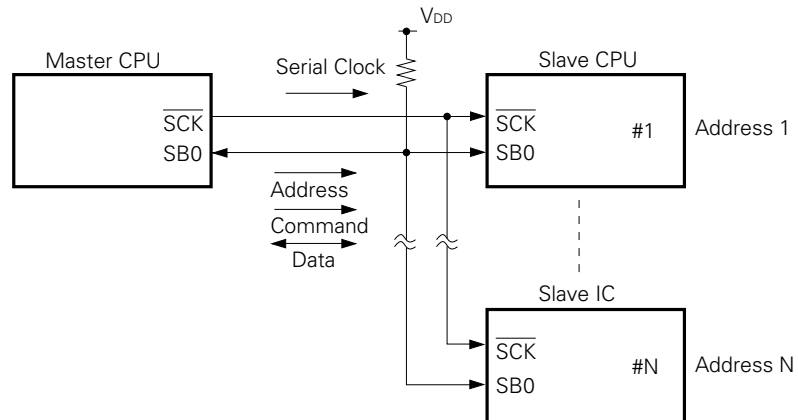
(3) SBI mode (serial bus interface mode)

In the SBI mode, communication is performed with multiple devices by means of two lines: The serial clock ($\overline{\text{SCK}}$) and the serial data bus (SB0).

This mode conforms to the NEC serial bus format.

In the SBI mode, the sender can output to the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. The receiver can determine by hardware whether the received data is an address, command or actual data.

Fig. 5-23 Example of SBI System Configuration

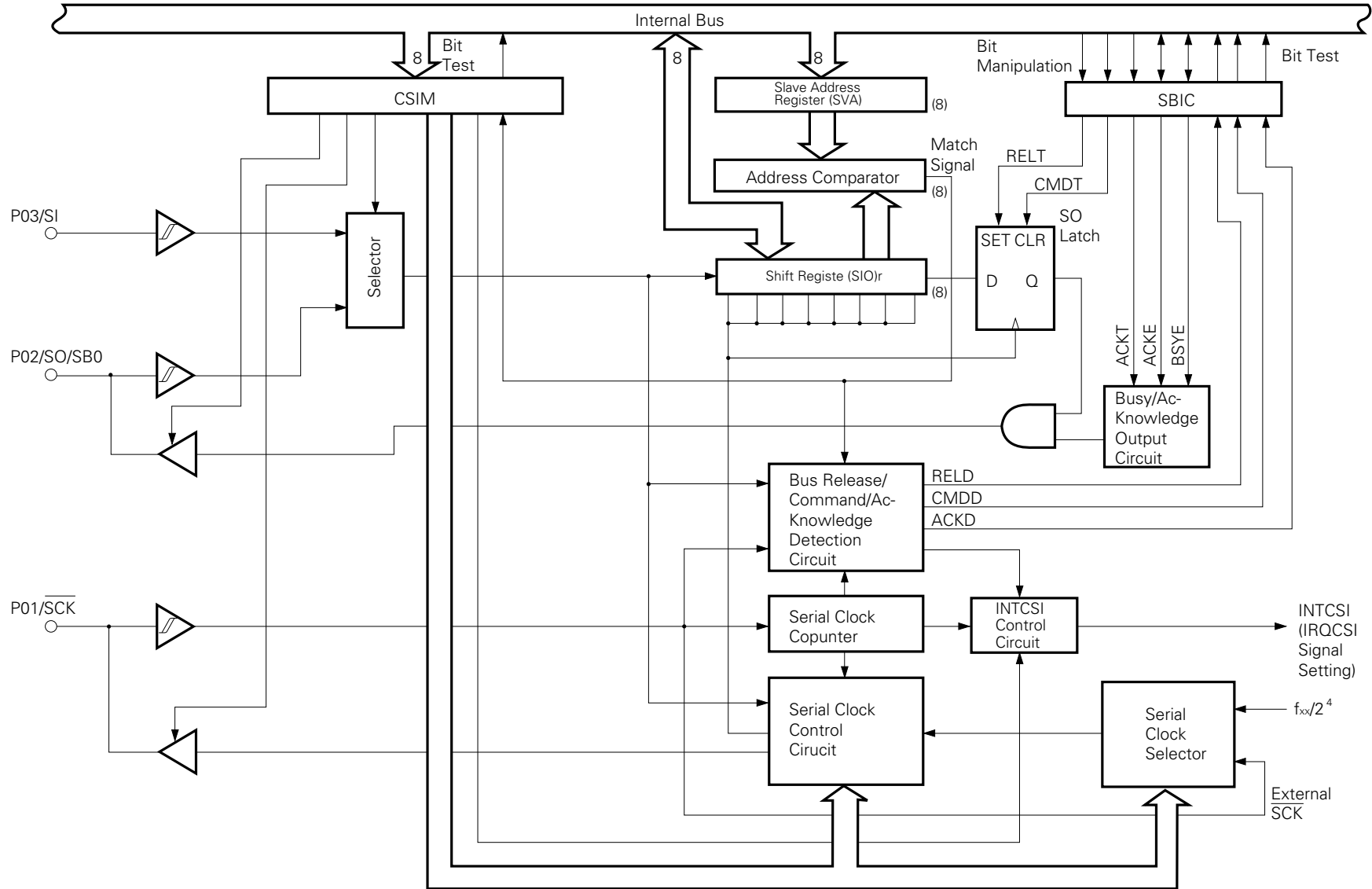


Note When the $\mu\text{PD75402A}$ is used as a slave CPU, its address is limited to the range C0H to C7H.

5.5.2 Serial Interface Configuration

The serial interface block diagram is shown in Fig. 5-24.

Fig. 5-24 Serial Interface Block Diagram



(1) Serial operating mode register (CSIM)

CSIM is an 8-bit register which specifies the serial interface operating mode, serial clock, wake-up function, etc. (See 5.5.3 (1) “Serial operating mode register” for details.)

(2) Serial bus interface control register (SBIC)

SBIC is an 8-bit register composed of bits which control the serial bus and flags which indicate various statuses of the input data from the serial bus, and is mainly used in the SBI mode. (See 5.5.3 (2) “Serial bus interface control register” for details.)

(3) Shift register (SIO)

The SIO register converts 8-bit serial data to parallel data and 8-bit parallel data to serial data. It performs transmission/reception operations (shift operations) in synchronization with the serial clock. A serial transfer is started by writing data to SIO. Actual transmission/reception operations are controlled by writes to the SIO. (See 5.5.3 (3) “Shift register” for details.)

(4) SO latch

A latch which holds the SO/SB0 and SI pin levels. Can be directly controlled by software. Set at the end of the 8th $\overline{\text{SCK}}$ pulse in the SBI mode. (See 5.5.3 (2) “Serial bus interface control register” for details.)

(5) Serial clock selector

Selects the serial clock to be used.

(6) Serial clock counter

Counts the serial clock cycles output and input in a transmission/reception operation, and checks that 8-bit data transmission/reception has been performed.

(7) Slave address register (SVA), address comparator

In SBI mode, this register is used when the $\mu\text{PD75402A}$ is used as a slave device. The slave sets its own specification number (slave address value) in the SVA register. The master outputs a slave address to select a specific slave.

The address comparator is used to compare the slave address received from the master with the SVA value: If they match the relevant slave is determined to have been selected. (See 5.5.3 (4) “Slave address register” for details.)

(8) INTCSI control circuit

Controls the generation of interrupt requests. In the following case, the interrupt requests (INTCSI) are generated and interrupt request flags (IRQCSI) are set (see **Fig. 6-1 “Interrupt Control Circuit Block Diagram”**).

- In 3-wire serial I/O mode
An interrupt request is generated on each count of 8 serial clock cycles.
 - In SBI mode
When WUP* = “0” ... An interrupt request is generated on each count of 8 serial clock cycles.
When WUP = “1” ... An interrupt request is generated when the SVA and SIO values match after address reception.
- * WUP: Wake-up function specified bit (bit 5 of CSIM)

(9) Serial clock control circuit

Controls the supply of the serial clock to the shift register. Also controls the clock output to the $\overline{\text{SCK}}$ pin when the internal system clock is used.

(10) Busy/acknowledge output circuit, bus release/command acknowledge detection circuit

These circuits perform output and detection of various control signals in the SBI mode.
They do not operate in the 3-wire serial I/O mode.

5.5.3 Register Functions**(1) Serial operating mode register (CSIM)**

The format of the serial operating mode register (CSIM) is shown in Fig. 5-25.

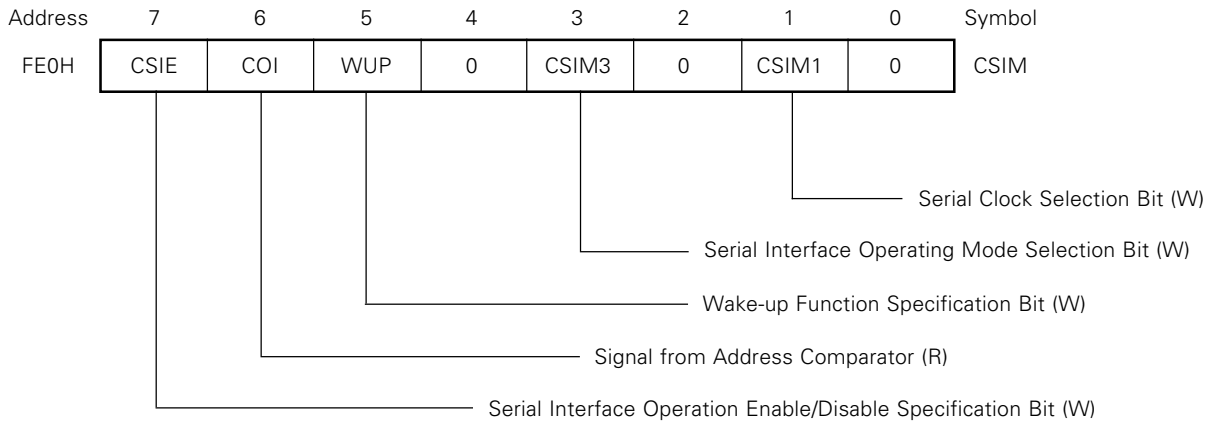
CSIM is an 8-bit register which specifies the serial interface operating mode, serial clock, wake-up function, etc.

CSIM is manipulated by 8-bit memory manipulation instructions. The high-order 3 bits can be manipulated bit by bit using the individual bit names.

Read/write capability differs from bit to bit (see **Fig. 5-25**). Bit 6 can be tested only, and data written to this bit is invalid.

Reset input clears this register to 00H.

Fig. 5-25 Serial Operating Mode Register (CSIM) Format (1/2)



Remarks (R) Read only
(W) Write only

Note 0 must be written to CSIM bits 4, 2, 0.

Serial clock selection bit (W)

CSIM1	Serial Clock		$\overline{\text{SCK}}$ Pin Mode
	3-Wire Serial I/O Mode	SBI Mode	
0	Input clock to $\overline{\text{SCK}}$ pin from off chip		Input
1	$f_{xx}/2^4$ (262 kHz)		Output

Remarks () When $f_{xx} = 4.19$ MHz

Serial interface operating mode selection bit (W)

CSIM3	Operating Mode	Shift Register Bit Order	SO Pin Function	SI Pin Function
0	3-wire serial I/O mode	SIO _{7 to 0} ↔ XA (MBS-first transfer)	SO/P02 (CMOS output)	SI/P03 (Input)
1	SBI mode	SIO _{7 to 0} ↔ XA (MBS-first transfer)	SB0/P02 (N-ch open-drain input/output)	P03 input

Fig. 5-25 Serial Operating Mode Register (CSIM) Format (2/2)

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer in each mode.
	1	Used only in SBI mode. IRQCSI is set only when the address received after bus release matches the slave address register data (wake-up status). SB0 is high impedance.

Note If WUP = 1 is set during $\overline{\text{BUSY}}$ signal output, $\overline{\text{BUSY}}$ is not released. With the SBI, the $\overline{\text{BUSY}}$ signal is output after the $\overline{\text{BUSY}}$ release directive until the next fall of the serial clock ($\overline{\text{SCK}}$). When setting WUP = 1, it is necessary to confirm that the SB0 pin has been driven high after $\overline{\text{BUSY}}$ is released before setting WUP = 1.

Signal from address comparator (R)

COI*	Clearing Condition (COI = 0)	Setting Condition (COI = 1)
	When slave address register (SVA) and shift register data do not match	When slave address register (SVA) and shift register data match.

* A COI read is valid only before the start or after completion of a serial transfer. During a transfer an indeterminate value will be read. Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI Pins
CSIE	0	Shift operation disabled	Cleared	Retained	Port 0 function only
	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

Remarks 1. The operating mode can be selected according to the setting of CSIE and CSIM3.

CSIE	CSIM3	Operating Mode
0	×	Operation-halted mode
1	0	3-wire serial I/O mode
1	1	SBI mode

2. The P10/ $\overline{\text{SCK}}$ pin status depends on the setting of CSIE and CSIM0 as shown below.

CSIE	CSIM1	P10/ $\overline{\text{SCK}}$ Pin Status
0	0	Input port
0	1	High-level output
1	0	Serial clock input (high impedance)
1	1	Serial clock output (high-level output)

3. The following procedure should be used to clear CSIE during a serial transfer.

- ① Clear the interrupt enable flag to set the interrupt disabled state.
- ② Clear CSIE.
- ③ Clear the interrupt request flag.

Example 1. This example selects $f_{xx}/2^4$ as the serial clock, generates an IRQCSI serial interrupt at the end of each serial transfer, and selects the mode in which serial transfers are performed in the SBI mode with the SB0 pin as the serial data bus.

```
MOV     XA, #10001010B
MOV     CSIM, XA      ; CSIM ← 10001010B
```

2. To enable serial transfers in accordance with the contents of CSIM.

```
SET1    CSIE
```


(2) Serial bus interface control register (SBIC)

The format of the serial bus interface control register (SBIC) is shown in Fig. 5-26.

SBIC is an 8-bit register composed of bits which control the serial bus and flags which indicate various statuses of the input data from the serial bus, and is mainly used in the SBI mode.

SBIC is manipulated by bit-manipulation instructions; it cannot be manipulated by 4-bit or 8-bit memory manipulation instructions.

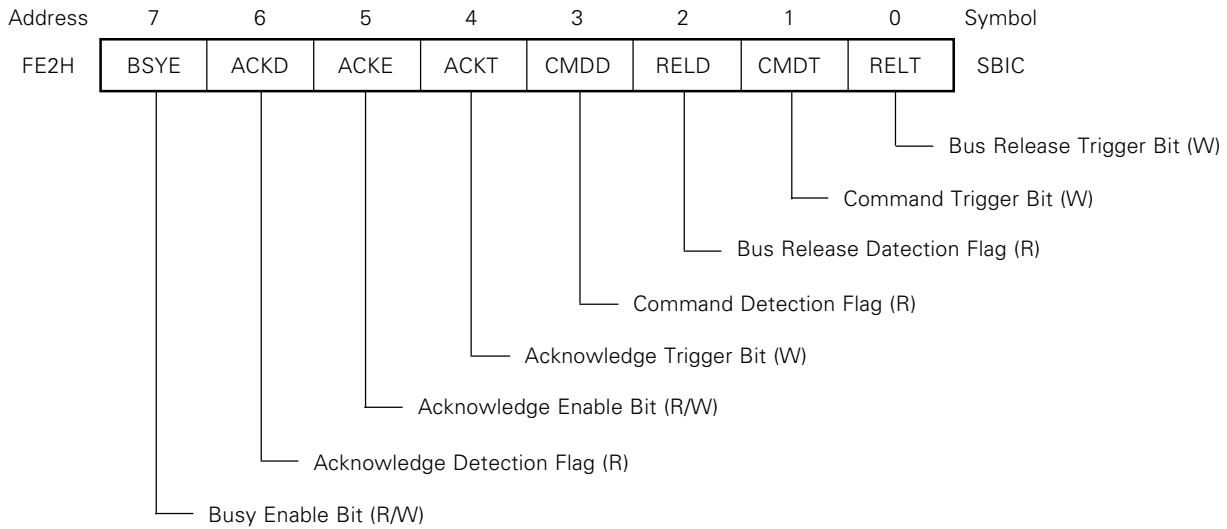
Read/write capability differs from bit to bit (see Fig. 5-26).

Reset input clears this register to 00H.

Note In the 3-wire serial I/O mode, only the following bits can be used:

- Bus release trigger bit (RELT) ... SO latch setting
- Command trigger bit (CMDT) ... SO latch clearing

Fig. 5-26 Serial Bus Interface Control Register (SBIC) Format (1/3)



Remarks (R) Read only
 (W) Write only
 (R/W) Read/write enabled

Fig. 5-26 Serial Bus Interface Control Register (SBIC) Format (2/3)

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).
------	---

Note SB0 must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Command trigger bit (W)

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).
------	---

Note SB0 must not be cleared during a serial transfer: Ensure that it is cleared before a transfer is started or after it is completed.

Bus release detection flag (R)

	Clearing Conditions (RELD = 0)	Setting Condition (RELD = 1)
RELD	<ul style="list-style-type: none"> ① When a transfer start instruction is executed ② When RESET is input ③ When CSIE = 0 (See Fig. 5-25) ④ When SVA and SIO do not match when an address is received 	When the bus release signal (REL) is detected

Command detection flag (R)

	Clearing Conditions (CMDD = 0)	Setting Condition (CMDD = 1)
CMDD	<ul style="list-style-type: none"> ① When a transfer start instruction is executed ② When the bus release signal (REL) is detected ③ When RESET is input ④ When CSIE = 0 (See Fig. 5-25) 	When the command signal (CMD) is detected

Acknowledge trigger bit (W)

ACKT	<p>When ACKT is set after the end of a transfer, \overline{ACK} is output in synchronization with the next SCK. After the \overline{ACK} signal is output, ACKT is automatically cleared (0).</p> <p>Note</p> <ul style="list-style-type: none"> 1. ACKT must not be set (1) before completion of a serial transfer or during a transfer. 2. ACKT cannot be cleared by software. 3. When ACKT is set, ACKE should be reset to 0.
------	---

Fig. 5-26 Serial Bus Interface Control Register (SBIC) Format (3/3)

Acknowledge enable bit (R/W)

ACKE	0	Disables automatic output of the acknowledge signal (\overline{ACK}) (output by ACKT is possible).	
	1	When set before end of transfer	\overline{ACK} is output in synchronization with the 9th SCK clock cycle.
When set after end of transfer		\overline{ACK} is output in synchronization with SCK immediately after execution of the setting instruction.	

Acknowledge detection flag (R)

ACKD	Clearing Conditions (ACKD = 0)		Setting Condition (ACKD = 1)
	①	When a transfer is started	When the acknowledge signal (\overline{ACK}) is detected (Synchronized with rise of SCK)
②	When RESET is input		

Busy enable bit (R/W)

BSYE	0	① Disabling of automatic busy signal output ② Busy signal output is stopped in synchronization with the fall of \overline{SCK} immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of \overline{SCK} following the acknowledge signal.

Example 1. To output the command signal.

```
SET1      CMDT
```

2. To test RELD and CMDD, and perform different processing according to the type of receive data. This interrupt routine is only performed when WUP = 1 and there is an address match.

```
SKF      RELD    ; Test RELD
BR       !ADRS
SKT      CMDD    ; Test CMDD
BR       !DATA
CMD      : ..... ; Command interpretation
DATA     : ..... ; Data processing
ADRS     : ..... ; Address decoding
```

(3) Shift register (SIO)

The configuration around the shift register is shown in Fig. 5-27. SIO is an 8-bit register which carries out parallel-to-serial conversion and performs serial transmission/reception (shift operations) in synchronization with the serial clock.

A serial transfer is started by writing data to SIO.

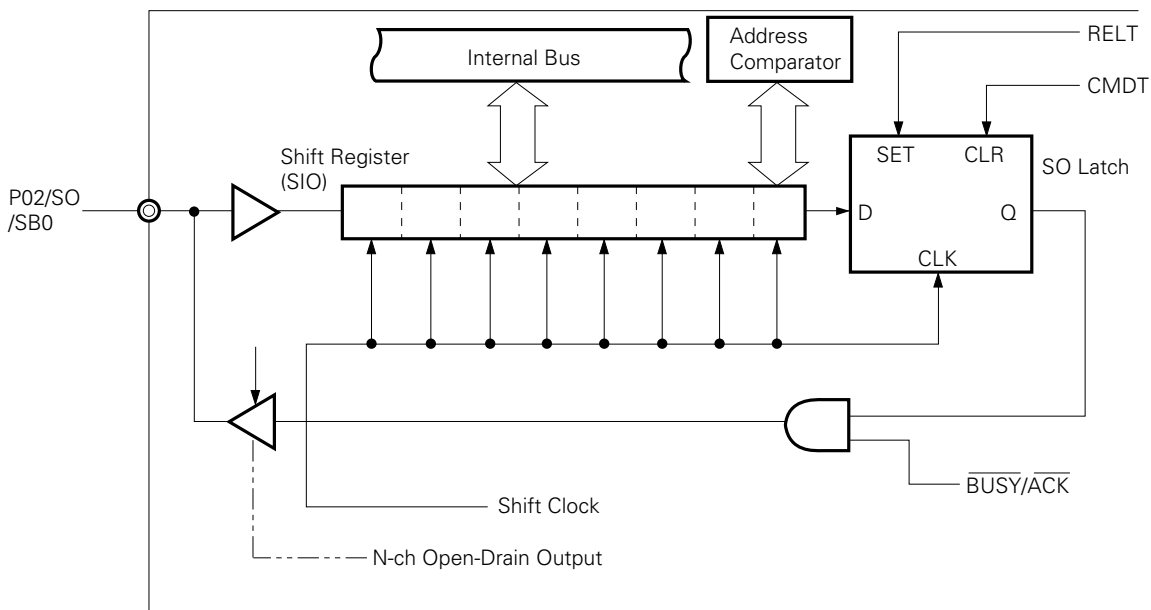
In transmission, the data written to SIO is output to the serial output (SO) or the serial data bus (SB0). In reception, data is read into SIO from the serial input (SI) or SB0.

SIO can be read or written to by an 8-bit manipulation instruction.

If RESET is input during its operation, the value of SIO is indeterminate. If RESET is input in the standby mode, the value of SIO is retained.

The shift operation stops after transmission/reception of 8 bits.

Fig. 5-27 Configuration Around Shift Register



SIO reading and the start of a serial transfer (write) are possible at the following times:

- When the serial interface operation enable/disable bit (CSIE) = 1, except when CSIE is set to "1" after data has been written into the shift register.
- When the serial clock has been masked after an 8-bit serial transfer.
- When \overline{SCK} is high.

Ensure that \overline{SCK} is high when data is written to or read from the SIO register.

In the SBI mode data bus configuration, input pins and output pins have dual functions. Output pins have an N-ch open-drain configuration. Therefore, in a device in which reception is about to be performed, FFH should be set in the SIO register.

(4) Slave address register (SVA)

SVA is an 8-bit register used by the slave to set the slave address value (its own specification number).

SVA is a write-only register which is manipulated by 8-bit manipulation instructions.

After RESET signal input, the value of SVA is indeterminate. However, when RESET is input in the standby mode, the value of SVA is retained.

In the SBI mode, the slave address register (SVA) has the following two functions:

(a) Slave address detection

Used when the μ PD75402A is connected to the serial bus as a slave device.

The high-order 5 bits of the SVA register are fixed at 11000 by hardware. Therefore, the address assigned to the μ PD75402A is limited to the range C0H to C7H.

The master outputs to its connected slaves a slave address to select a specific slave. If these two data items (the slave address output from the master and the SVA value) are found to match when compared by the address comparator, the relevant slave is determined to have been selected.

At this time, bit 6 (COI) of the serial operating mode register (CSIM) is set to "1".

When an address is received the bus release detection flag (RELD) is cleared (0) if a match is not detected. IRQCSI is set only when a match is detected when WUP = 1. This interrupt request indicates that a communication request has been issued from the master to the μ PD75402A.

(b) Error detection

The SVA performs error detection in the following cases:

- When the μ PD75402A transmits addresses, commands or data as the master device.
- When the μ PD75402A transmits data as a slave device.

See 5.5.6 (8) "Error detection" for details.

Note When data is written to SVA, ensure that a value between C0H and C7H is written.

5.5.4 Operation-Halted Mode

The operation-halted mode is used when no serial transfer is performed, allowing power dissipation to be reduced.

In this mode, the shift register does not perform shift operations and can be used as an ordinary 8-bit register.

When the RESET signal is input the operation-halted mode is set. The P02/SO/SB0 and P03/SI pins are fixed as input ports. P01/ $\overline{\text{SCK}}$ can be used as an input port depending on the setting of the serial operating mode register.

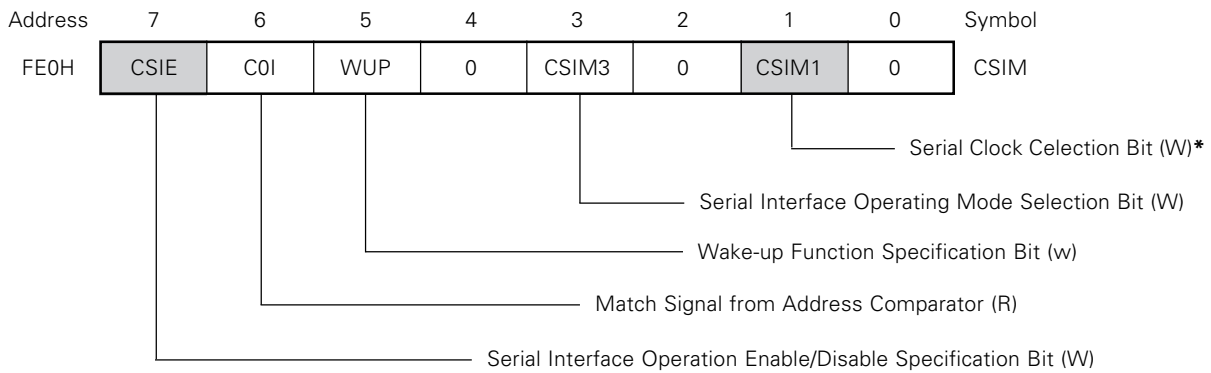
(1) Register setting

Operation-halted mode setting is performed by the serial operating mode register (CSIM) (see 5.5.3 (1) "Serial operating mode register" for full details of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions, but bit manipulation of CSIE is also possible. Manipulation is also possible using bit names.

Reset input clears this register to 00H.

The shaded area indicates bits used in the operation-halted mode.



* Allow selection of P01/ $\overline{\text{SCK}}$ pin status.

Remarks (R) Read only
 (W) Write only

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI Pins
		Shift operation disabled	Cleared	Retained	Port o function only
CSIE	0				

Serial clock selection bit (W)

The P01/ $\overline{\text{SCK}}$ pin status depends on the CSIM1 setting as shown below.

CSIM1	P01/ $\overline{\text{SCK}}$ Pin Status
0	High impedance
1	High level

The following procedure should be used to clear CSIE during a serial transfer.

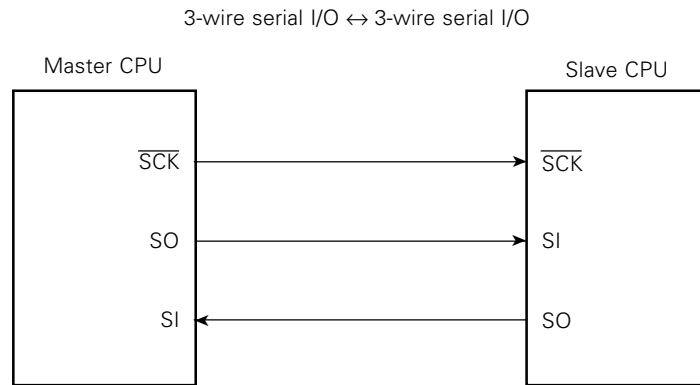
- ① Clear the interrupt enable flag (IECSI) to set the interrupt disabled state.
- ② Clear CSIE.
- ③ Clear the interrupt request flag (IRQCSI).

5.5.5 3-Wire Serial I/O Mode Operation

The 3-wire serial I/O mode allows connection to the system used in the 75X series, μPD7500 series, 87AD series, etc.

Communication is performed using three lines: The serial clock ($\overline{\text{SCK}}$), serial output (SO), and serial input (SI).

Fig. 5-28 Example of 3-Wire Serial I/O System Configuration



(1) Register setting

When the device is used in the 3-wire serial I/O mode, setting can be performed by means of the following two registers:

- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

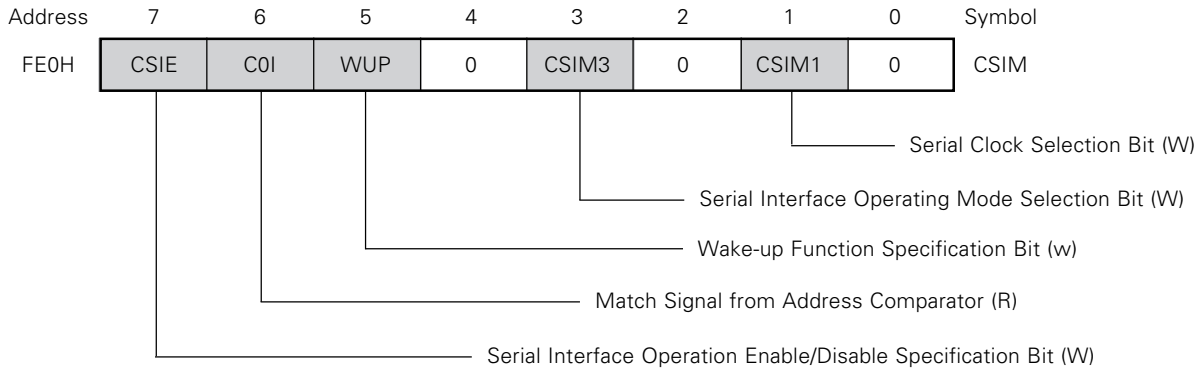
(a) Serial operating mode register (CSIM)

When the 3-wire serial I/O mode is used, CSIM is set as shown below (see 5.5.3 (1) “Serial operating mode register” for full details of CSIM).

CSIM is manipulated by 8-bit memory manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible.

Reset input clears the CSIM register to 00H.

The shaded area indicates bits used in the 3-wire serial I/O mode.



Remarks (R) Read only
(W) Write only

Serial clock selection bit (W)

CSIM1	Serial Clock		$\overline{\text{SCK}}$ Pin Mode
	3-Wire Serial I/O Mode	SBI Mode	
0	Input clock to $\overline{\text{SCK}}$ pin from off chip		Input
1	$f_{xx}/2^4$ (262 kHz)		Output

Remarks Figuer in () apply to $f_{xx} = 4.19$ MHz operation

Serial interface operating mode selection bit (W)

CSIM3	Operating Mode	Shift Register Bit Order	SO Pin Function	Si Pin Function
0	3-wire serial I/O mode	SIO _{7 to 0} ↔ XA (MSB-fist transfer)	SO/P02 (CMOS Output)	SI/P03 (Input)

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer.
-----	---	---

Signal from address comparator (R)

COI*	Clearing Conditions (COI = 0)	Setting Condition (COI = 1)
	When slave address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

* A CIO read is valid only before the start of after completion of a serial transfer. During a transfer an indeterminate value will be read.

Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI Pins
CSIE	1	Shift operation enabled	Count operation	Settable	Function in each mode plus port 0 function

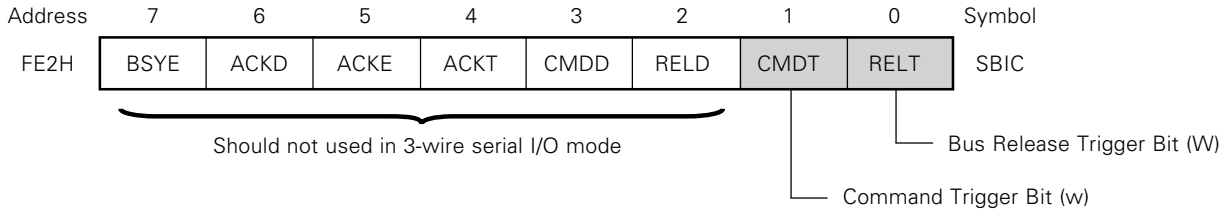
(b) Serial bus interface control register (SBIC)

When the 3-wire serial I/O mode is used, SBIC is set as shown below (see 5.5.3 (2) “Serial bus interface control register” for full details of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

The shaded area indicates bits used in the 3-wire serial I/O mode.



Remarks (W) Write only

Bus release trigger bit (W)

RELT	The command signal (REL) trigger output control bit. The SO latch is cleared (0) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).
------	---

Command trigger bit

CMDT	The command signal (CMD) Trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).
------	---

Note Bits other than RELT and CMDT should not be used in the 3-wire serial I/O mode.

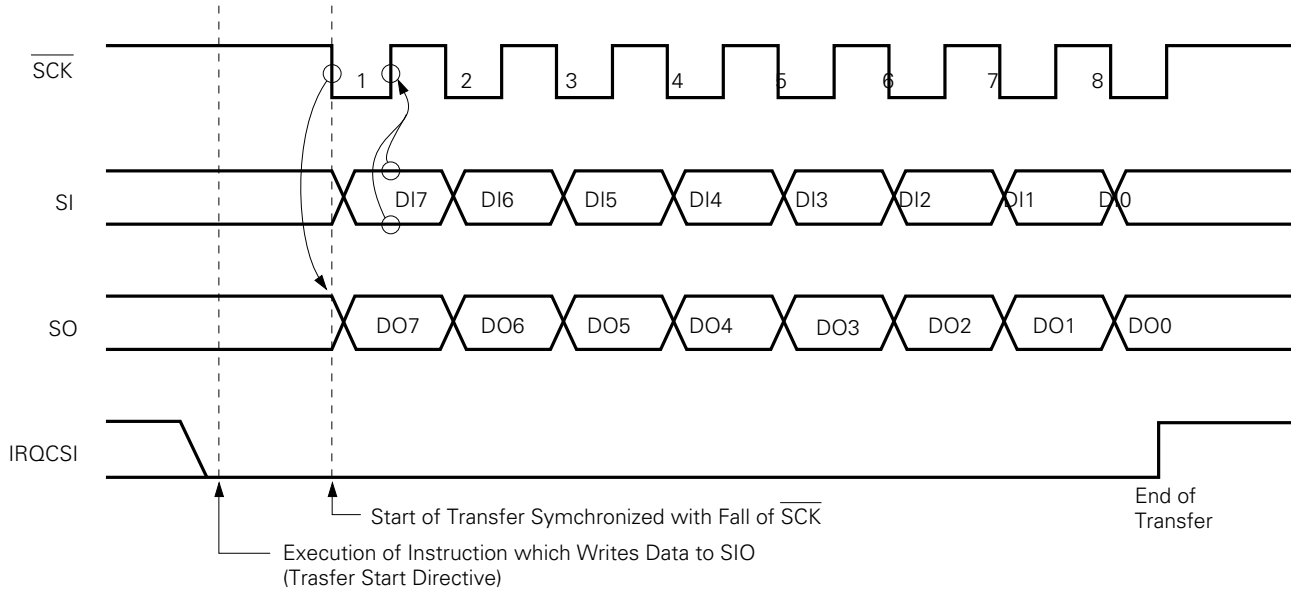
(2) Communication operation

In the 3-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in synchronization with the serial clock.

Shift register shift operations are performed in synchronization with the fall of the serial clock (\overline{SCK}). Then send data is held in the SO latch output from the SO pin. Also, receive data input to the SI pin is latched in the shift register on the rise of \overline{SCK} .

At the end of an 8-bit transfer the operation of the shift register stops automatically and the IRQCSI interrupt request flag is set.

Fig. 5-29 3-Wire Serial I/O Mode Timing



The SO pin becomes a CMOS output and outputs the SO latch status, and thus the SO pin output status can be manipulated in accordance with the setting of the RELT bit and CMDT bit.

However, manipulation should not be performed during a serial transfer.

(3) Serial clock selection

Serial clock selection is performed by setting bit 1 of the serial operating mode register (CSIM). Either of the following clocks can be selected.

Table 5-6 Serial Clock Selection and Use (in 3-Wire Serial I/O Mode)

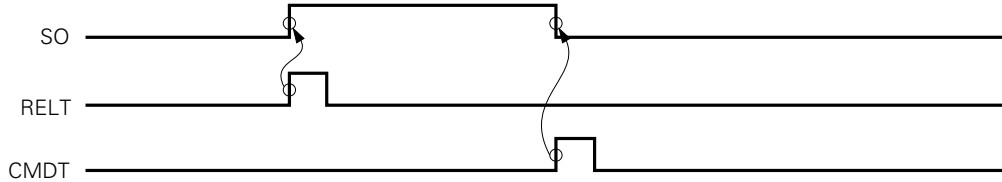
Mode Register	Serial Clock		Possible Timing for Shift Register R/W and Serial Transfer Start	Use
	Source	Serial Clock Masking		
0	External SCK	Automatically masked at end of 8-bit data transfer.	Possible only when serial transfer is halted* or when SCK is high.	Slave CPU
1	$f_{xx}/2^4$		Possible only when serial transfer is halted* or when SCK is high.	Medium-speed serial transfer

* "When serial transfer is halted" means in the operation-halted mode, or when the serial clock is masked after an 8-bit transfer.

(4) Signals

RELT and CMDT operation is shown in Fig. 5-30.

Fig. 5-30 RELT & CMDT Operation

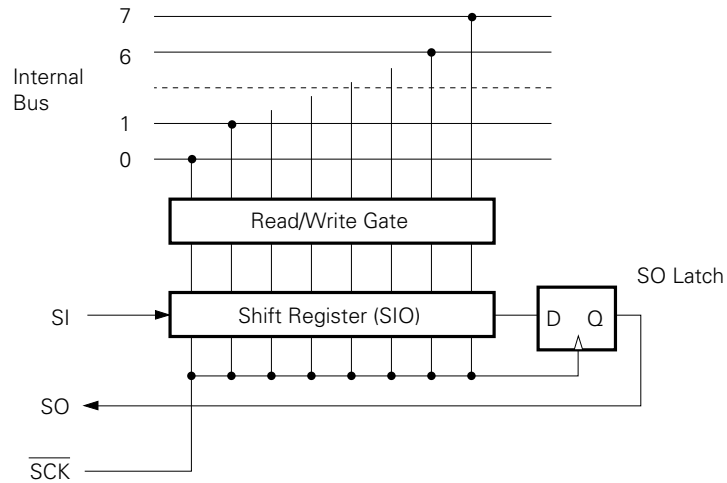


(5) Data transfer order

The μ PD75402A 3-wire serial I/O mode differs from that of other 75X series products in that it is not possible to switch between MSB and LSB as the first bit.

Serial transfer is performed MSB-first.

Fig. 5-31 Shift Register (SIO) and Internal Bus Configuration



(6) Start of transfer

When the following two conditions are met a serial transfer is started by setting transfer data in the shift register (SIO).

- The serial interface operation enable/disable bit (CSIE) = 1.
- After an 8-bit serial transfer, the internal serial clock is stopped or $\overline{\text{SCK}}$ is high.

Note The transfer will not be started if CSIE is set to "1" after data is written into the shift register.

When an 8-bit transfer ends, the serial transfer stops automatically and the IRQCSI interrupt request flag is set.

Example In the following example the data in the RAM specified by the HL register is transferred to SIO, and at the same time the SIO data is fetched into the accumulator and the serial transfer is started.

```
MOV  XA, @HL ; Fetch send data from RAM
XCH  XA, SIO ; Exchange send data with receive data and start transfer
```

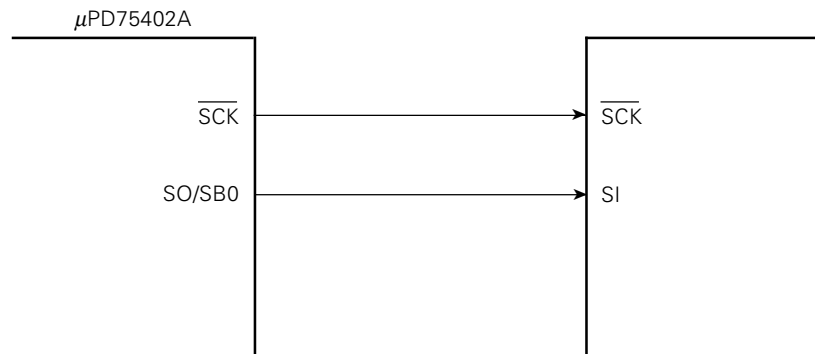
(7) 3-wire serial I/O mode applications

(a) To transfer data MSB-first (master operation) using a 262 kHz transfer clock (when operating at 4.19 MHz).

<Sample program>

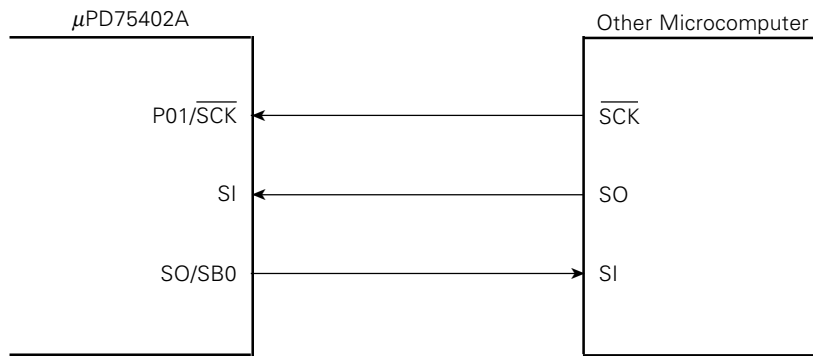
```
MOV  XA, #10000010B
MOV  CSIM, XA      ; Transfer mode setting
MOV  XA, TDATA     ; TDATA is transfer data storage address
MOV  SIO, XA       ; Transfer data setting & start of transfer
```

Note From the second time onward, the transfer can be started by setting data in SIO (MOV SIO, XA or XCH XA, SIO).



In this application the μ PD75402A SI pin can be used as an input port.

(b) To transmit/receive MSB-first data using an external clock (slave operation).



<Sample program>

Main routine

```

MOV  XA, #80H
MOV  CSIM, XA    ; Serial operation stopped, external clock specification
MOV  XA, TDATA
MOV  SIO, XA    ; Transfer data setting & start of transfer
EI   IECSI
EI
  
```

Interrupt routine

```

MOV  XA, TDATA
XCH  XA, SIO    ; Receive data ÷ send data & start of transfer
MOV  RDATA, XA ; Receive data save
RETI
  
```

5.5.6 SBI Mode Operation

The SBI (serial bus interface) is a high-speed serial interface which conforms to the the NEC serial bus format.

The SBI is a single-master high-speed serial bus. Its format includes the addition of bus configuration functions to the clocked serial I/O method to enable communication to be performed with multiple devices using two signal lines. Consequently, when a serial bus is configured with multiple microcomputers and peripheral ICs, it is possible to reduce the number of ports used and the amount of wiring on the substrate.

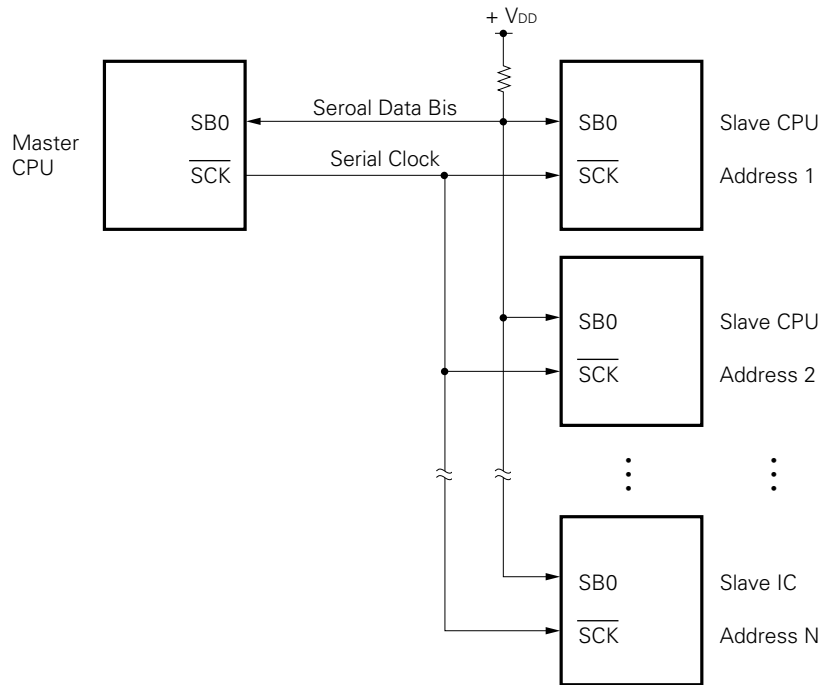
The master can output to a slave on the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. The slave can determine by hardware whether the received data is an address, command or actual data. This function allows the serial interface control portion of the application program to be simplified.

SBI functions are incorporated in a number of devices including the 75X series, and 78K series 8-bit single-chip microcomputers.

An example of a serial bus configuration when CPUs and peripheral ICs with a serial interface conforming to the SBI are used is shown in Fig. 5-32.

In the SBI the SB0 serial data bus pin is an open-drain output and thus the serial data bus line is in the wired-OR state. The serial data bus line requires a pull-up resistor.

Fig. 5-32 Example of SBI Serial Bus System Configuration



Note When master/slave exchange processing is performed, since serial clock line ($\overline{\text{SCK}}$) input/output switching is performed asynchronously between master and slave, a pull-up resistor is also required for the serial clock line ($\overline{\text{SCK}}$).

(1) SBI functions

Since conventional serial I/O methods have only data transfer functions, when a serial bus is configured with multiple devices connected a large number of ports and wires are required for Chip Select signal and command/data differentiation, busy status recognition, etc. If these controls are performed by software, the load incurred by software is very large.

When the SBI, a serial bus can be configured using only two lines: The serial clock, \overline{SCK} , and the serial data bus, SB0. As a result, the number of microcomputer ports and the amount of substrate wiring can be significantly reduced.

SBI functions are described below.

(a) Address/command/data differentiation function

Identifies serial data as an address, command or actual data.

(b) Chip selection by address

The master performs chip selection by address transmission.

(c) Wake-up function

A slave can identify address reception (chip selection) easily by means of the wake-up function (settable/releasable by software).

When the wake-up function is set, an interrupt (IRQCSI) is generated when a matching address is received.

As a result, non-selected CPUs can operate without regard to serial communications even when communication with multiple devices is performed.

(d) Acknowledge signal (\overline{ACK}) control function

Controls the acknowledge signal used to confirm serial data reception.

(e) Busy signal (\overline{BUSY}) control function

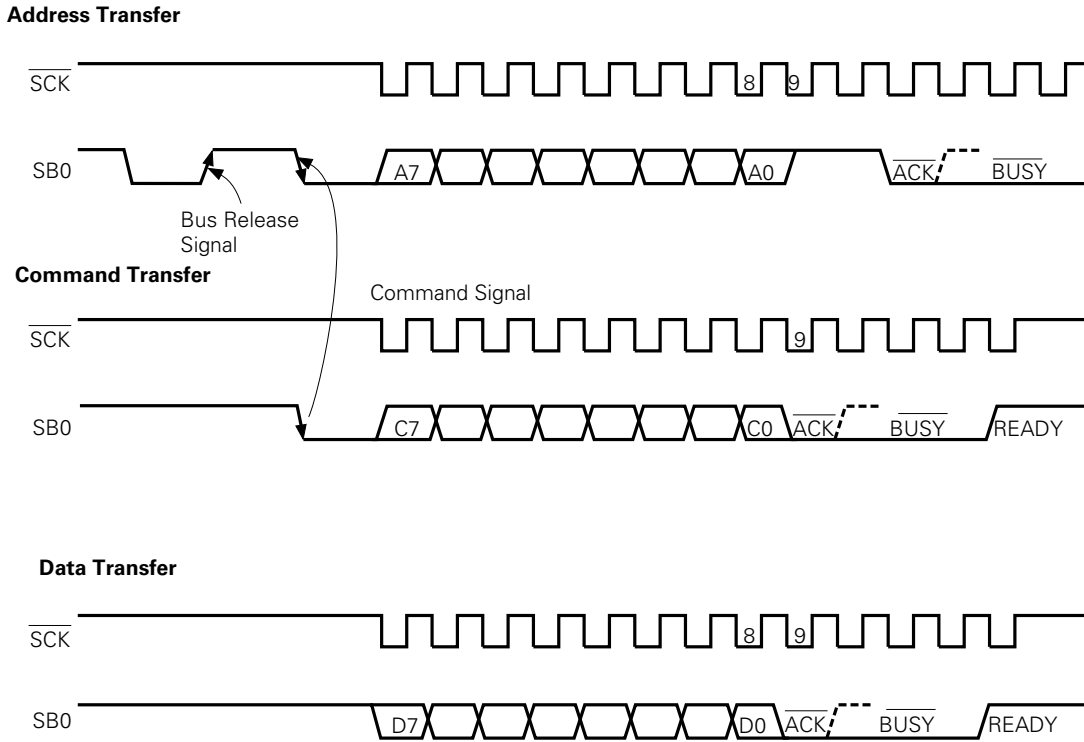
Controls the busy signal used to give notification of a slave busy status.

(2) SBI definition

The SBI serial data format and the meaning of the signals used are explained in the following section. Serial data transmitted via the SBI is classified into three types: Commands, addresses and data. Serial data forms a frame with the configuration shown below.

Address, command and data transfer timing is shown in Fig. 5-33.

Fig. 5-33 SBI Transfer Timing



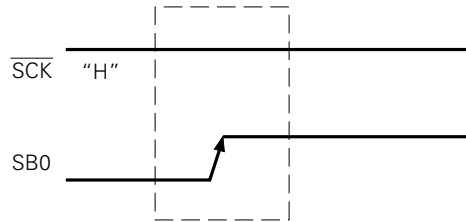
The bus release signal and command signal are output by the master. The \overline{BUSY} signal is output by the slave. \overline{ACK} can be output by either the master or slave (normally output by the 8-bit data receiver).

The serial clock is output by the master continuously from the start of an 8-bit data transfer until \overline{BUSY} is released.

(a) Bus release signal (REL)

The bus release signal indicates that the SB0 line has changed from low to high when the $\overline{\text{SCK}}$ line is high (when the serial clock is not being output). This signal is output by the master.

Fig. 5-34 Bus Release Signal

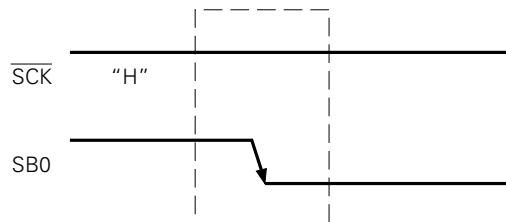


The bus release signal indicates that the master is about to send an address to a slave. Slaves incorporate hardware to detect the bus release signal.

(b) Command signal (CMD)

The command signal indicates that the SB0 line has changed from high to low when the $\overline{\text{SCK}}$ line is high (when the serial clock is not being output). This signal is output by the master.

Fig. 5-35 Command Signal

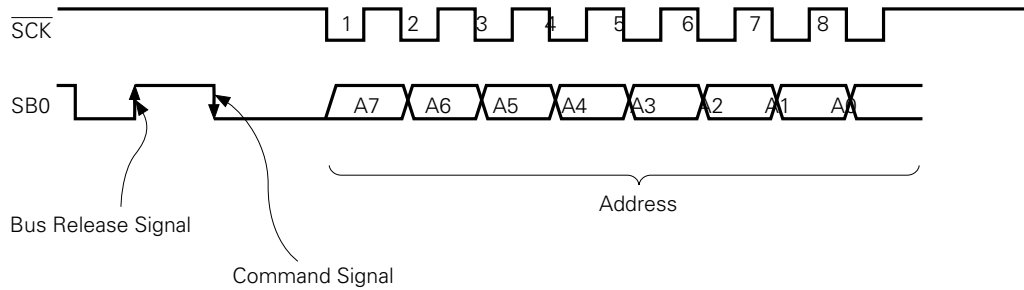


Slave incorporate hardware to detect the command signal.

(c) Address

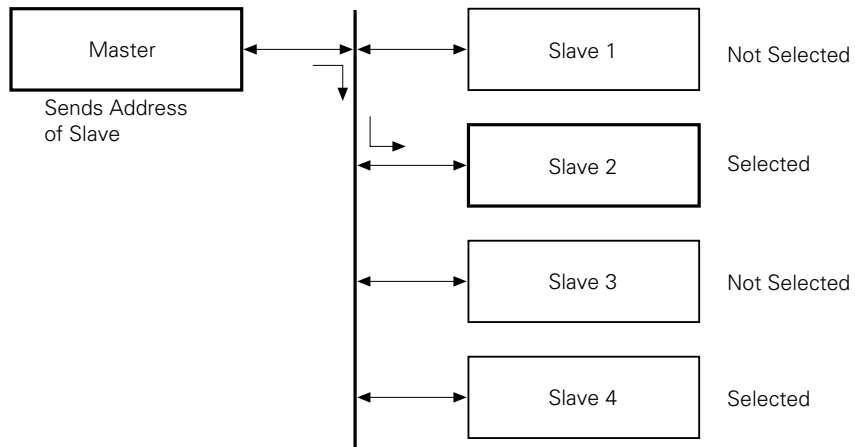
An address is 8-bit data output by the master to slaves connected to the bus line in order to select a particular slave.

Fig. 5-36 Address



The 8-bit data following the bus release signal and command signal is defined as an address. In a slave this condition is detected by hardware and a check is performed by hardware to see if the 8-bit data matches the slave's own specification number (slave address). If the 8-bit data matches the slave address, that slave is determined to have been selected and communication is subsequently performed with the master until a disconnect directive is received from the master.

Fig. 5-37 Slave Selection by Address



(d) **Command & data**

The master performs command transmission to or data transmission/reception to/from the slave selected by address transmission.

Fig. 5-38 Command

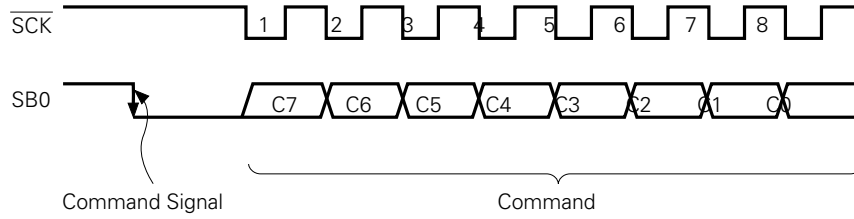
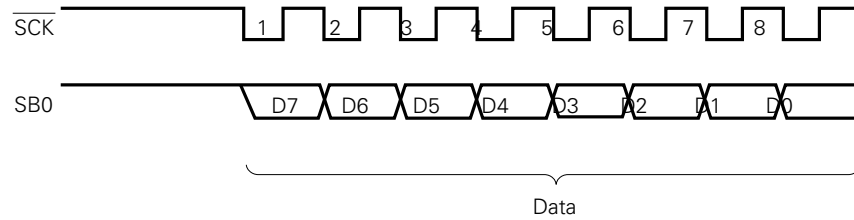


Fig. 5-39 Data

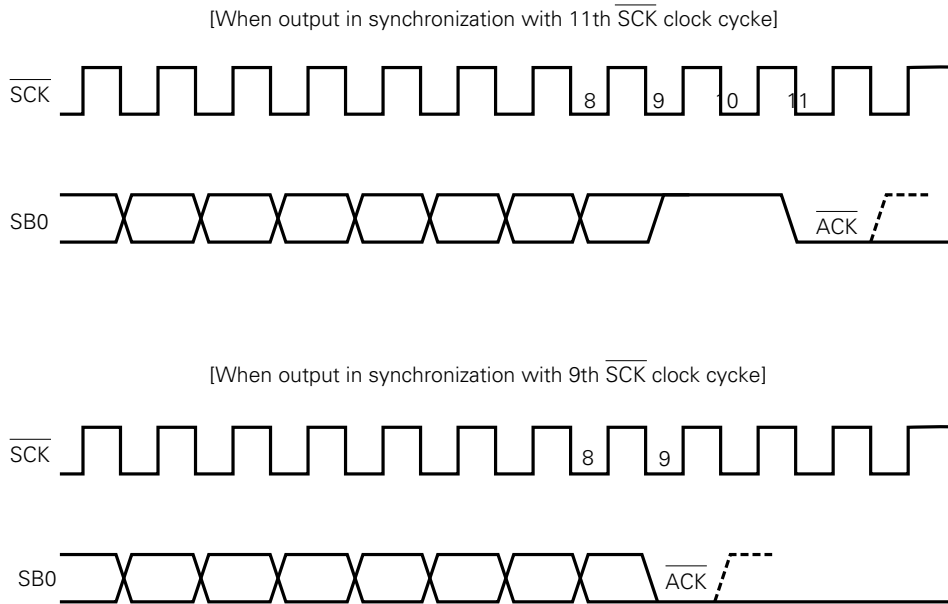


The 8-bit data following the command signal is defined as a command. 8-bit data with no command signal is defined as data. The way in which commands and data are used can be freely decided according to the communication specifications.

(e) Acknowledge signal ($\overline{\text{ACK}}$)

The acknowledge signal is used to confirm serial data reception between the sender and receiver.

Fig. 5-40 Acknowledge Signal



The acknowledge signal is a one-shot pulse synchronized with the fall of $\overline{\text{SCK}}$ after an 8-bit data transfer. Its position is arbitrary and it can be synchronized with any $\overline{\text{SCK}}$ clock cycle.

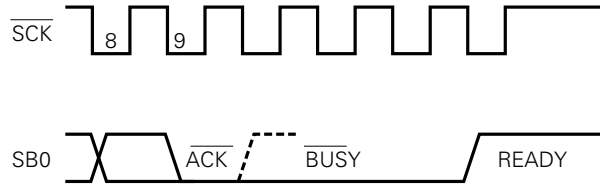
After 8-bit data transmission the sender checks whether the receiver has sent back an acknowledge signal. If an acknowledge signal is not returned within a specific time after data transmission, reception can be judged not to have been performed correctly.

(f) **Busy signal ($\overline{\text{BUSY}}$), ready signal ($\overline{\text{READY}}$)**

The busy signal notifies the master that a slave is preparing for data transmission/reception.

The ready signal notifies the master that a slave is ready for data transmission/reception.

Fig. 5-41 Busy Signal & Ready Signal



With the SBI a slave reports its busy status to the master by driving the SB0 line low.

The busy signal is output following the acknowledge signal output by the master or slave. Busy signal setting/release is performed in synchronization with the fall of $\overline{\text{SCK}}$. When the busy signal is released the master automatically terminates output of the $\overline{\text{SCK}}$ serial clock.

When the busy signal is released and the ready signal state is entered the master can start the next transfer.

(3) Register setting

When the device is used in the SBI mode, setting can be performed by means of the following two registers:

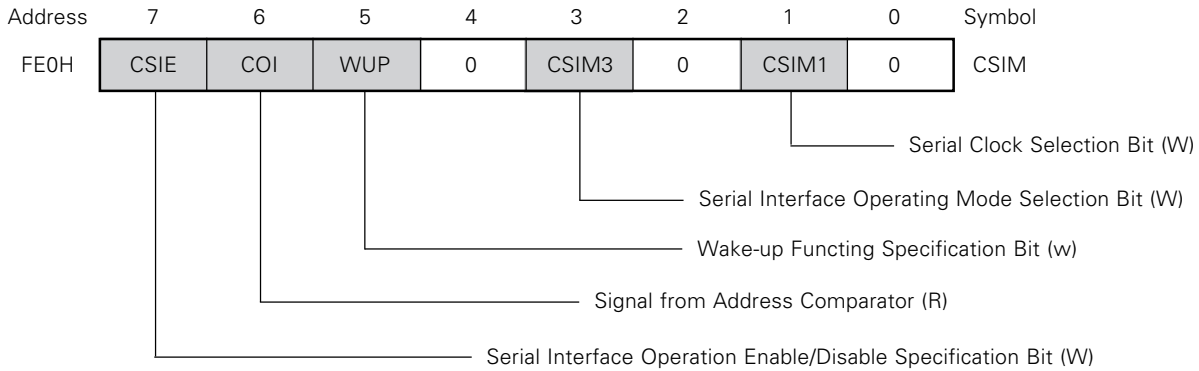
- Serial operating mode register (CSIM)
- Serial bus interface control register (SBIC)

(a) Serial operating mode register (CSIM)

When the SBI mode is used, CSIM is set as shown below (see 5.5.3 (1) “Serial operating mode register” for full details of CSIM).

CSIM is manipulated by 8-bit manipulation instructions. Bit manipulation of bits 7, 6 and 5 is also possible. Reset input clears the CSIM register to 00H.

The shaded area indicates bits used in the SBI mode.



Remarks (R) Read only
(W) Write only

Serial clock selection bit (W)

CSIM1	Serial Clock		$\overline{\text{SCK}}$ Pin Mode
	3-Wire Serial I/O Mode	SBI Mode	
0	Input clock to $\overline{\text{SCK}}$ pin from off chip		Input
1	$f_{xx}/2^4$ (262 kHz)		Output

Remarks Figure in () apply to $f_{xx} = 4.19$ MHz operation

Serial interface operating mode selection bit (W)

CSIM3	Operating Mode	Shift Register Bit Order	SO Pin Function	SI Pin Function
1	SBI mode	SIO _{7 to 0} ↔ XA (MBS-first transfer)	SB0/P02 (N-ch open-drain input/output)	P03 input

Wake-up function specification bit (W)

WUP	0	IRQCSI set at end of every serial transfer in SBI mode mask state.
	1	User only when functioning as a slave in SBI mode. IRQCSI is set only when the address received after bus release matches the slave address register data (wake-up status). SB0 is high impedance.

Note If $WUP = 1$ is set during \overline{BUSY} signal output, \overline{BUSY} is not released. With the SBI, the \overline{BUSY} signal is output after the \overline{BUSY} release directive until the next fall of the serial clock (\overline{SCK}). when setting $WUP = 1$, it is necessary to confirm that the SB0 pin has been driven high after \overline{BUSY} is released before setting $WUP = 1$.

Signal from address comparator (R)

COI*	Clearing Conditions (COI = 0)	Setting Condition (COI = 1)
	When slave address register (SVA) and shift register data do not match.	When slave address register (SVA) and shift register data match.

* A COI read is valid only before the start or after completion of a serial transfer. During a transfer an indeterminate value will be read.

Also, COI data written by an 8-bit manipulation instruction is ignored.

Serial interface operation enable/disable specification bit (W)

		Shift Register Operation	Serial Clock Counter	IRQCSI Flag	SO/SB0 & SI Pins
		CSIE	1	Shift operation enabled	Count operation

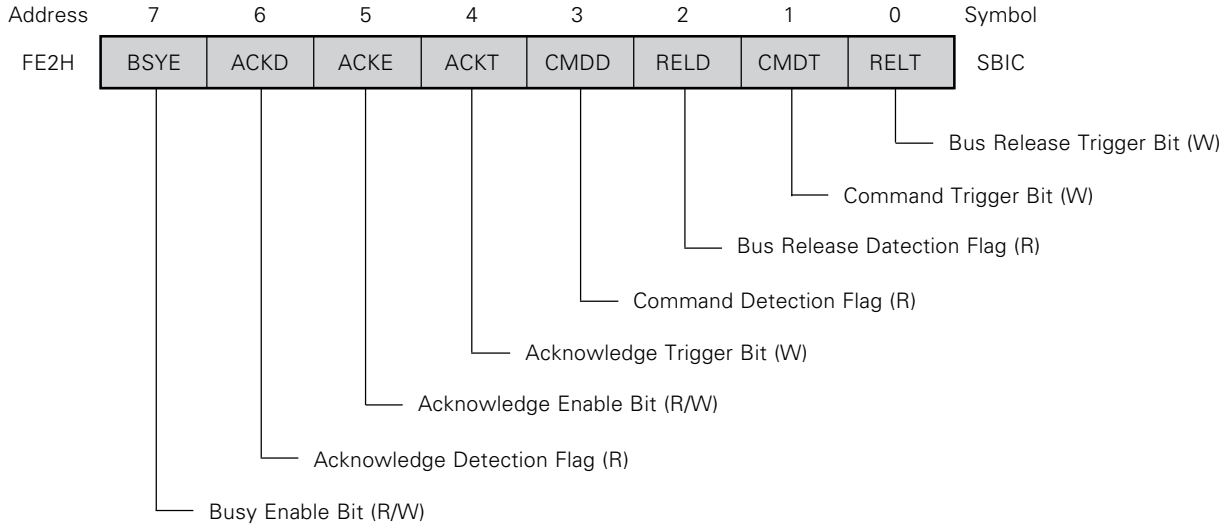
(b) Serial bus interface control register (SBIC)

When the SBI mode is used, SBIC is set as shown below (see 5.5.3 (2) “Serial bus interface control register” for full details of SBIC).

SBIC is manipulated by bit manipulation instructions.

Reset input clears the SBIC register to 00H.

The shaded area indicates bits used in the SBI mode.



- Remarks**
- (R) Read only
 - (W) Write only
 - (R/W) Read/write enabled

Bus release trigger bit (W)

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit (RELT = 1), after which the RELT bit is automatically cleared (0).
------	---

Note SB0 must not be set during a serial transfer: Ensure that it is set before a transfer is started or after it is completed.

Command trigger bit (W)

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit (CMDT = 1), after which the CMDT bit is automatically cleared (0).
------	---

Note SB0 must not be set during a serial transfer: Ensure that it is set before a transfer is started or after it is completed.

Bus release detection flag (R)

	Clearing Conditions (RELD = 0)	Setting Condition (RELD = 1)
RELD	<ul style="list-style-type: none"> ① When a transfer start instruction is executed ② When RESET is input ③ When CSIE = 0 (See Fig. 5-25) ④ When SVA and SIO do not match when an address is received 	When the bus release signal (REL) is detected

Command detection flag (R)

	Clearing Conditions (CMDD = 0)	Setting Condition (CMDD = 1)
CMDD	<ul style="list-style-type: none"> ① When a transfer start instruction is executed ② When the bus release signal (REL) is detected ③ When RESET is input ④ When CSIE = 0 (See Fig. 5-25) 	When the command signal (CMD) is detected

Acknowledge trigger bit (W)

ACKT	<p>When ACKT is set after the end of a transfer, \overline{ACK} is output in synchronization with the next SCK. After the \overline{ACK} signal is output, ACKT is automatically cleared (0).</p> <p>Note 1. ACKT must not be set (1) before completion of a serial transfer or during a transfer. 2. ACKT cannot be cleared by software. 3. When ACKT is set, ACKE should be reset to 0.</p>
------	---

Acknowledge enable bit (R/W)

ACKE	0	Disables automatic output of the acknowledge signal (output by ACKT is possible).	
	1	When set before end of transfer	\overline{ACK} is output in synchronization with the 9th SCK clock cycle.
When set after end of transfer		\overline{ACK} is output in synchronization with SCK immediately after execution of the setting instruction.	

Acknowledge detection flag (R)

	Clearing Conditions (ACKD = 0)	Setting Condition (ACKD = 1)
ACKD	<ul style="list-style-type: none"> ① When a transfer is started ② When RESET is input 	When the acknowledge signal (\overline{ACK}) is detected (Synchronized with the rise of SCK)

Busy enable bit (R/W)

BSYE	0	① Disabling of automatic busy signal output ② Busy signal output is stopped in synchronization with the fall of \overline{SCK} immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of \overline{SCK} following the acknowledge signal.

(4) Serial clock selection

Serial clock selection is performed by setting bit 1 of the serial operating mode register (CSIM). Either of the following clocks can be selected.

Table 5-7 Serial Clock Selection and Use (in SBI Mode)

Mode Register	Serial Clock		Possible Timing for Shift Register R/W and Serial Transfer Start	Use
	CSIM 1	Source		
0	External \overline{SCK}	Automatically masked at end of 8-bit data transfer.	Possible only when serial transfer is halted* or when \overline{SCK} is high.	Slave CPU
1	$f_{xx}/2^4$		Possible only when serial transfer is halted* or when \overline{SCK} is high.	Medium-speed serial transfer

* "When serial transfer is halted" means in the operation- halted mode, or when the serial clock is masked after an 8-bit transfer.

When the internal system clock is selected \overline{SCK} stops at 8 pulses internally, but externally the count continues until the slave is in the ready state.

(5) Signals

The operation of signals and flags in SBIC in the SBI mode are shown in Figs. 5-42 to 5-47, and SBI signals are listed in Table 5-8.

Fig. 5-42 RELT, CMDT, RELD & CMDD Operation (Master)

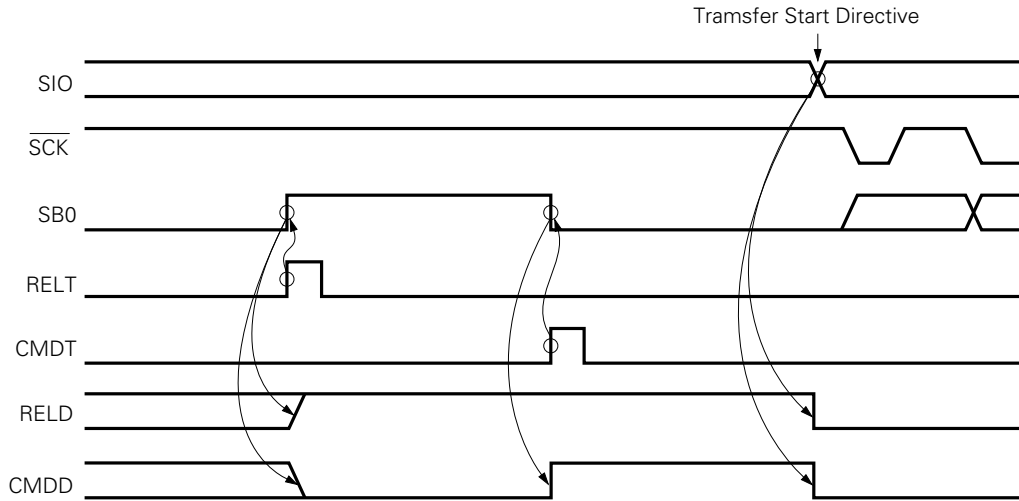


Fig. 5-43 RELT, CMDT, RELD & CMDD Operation (Slave)

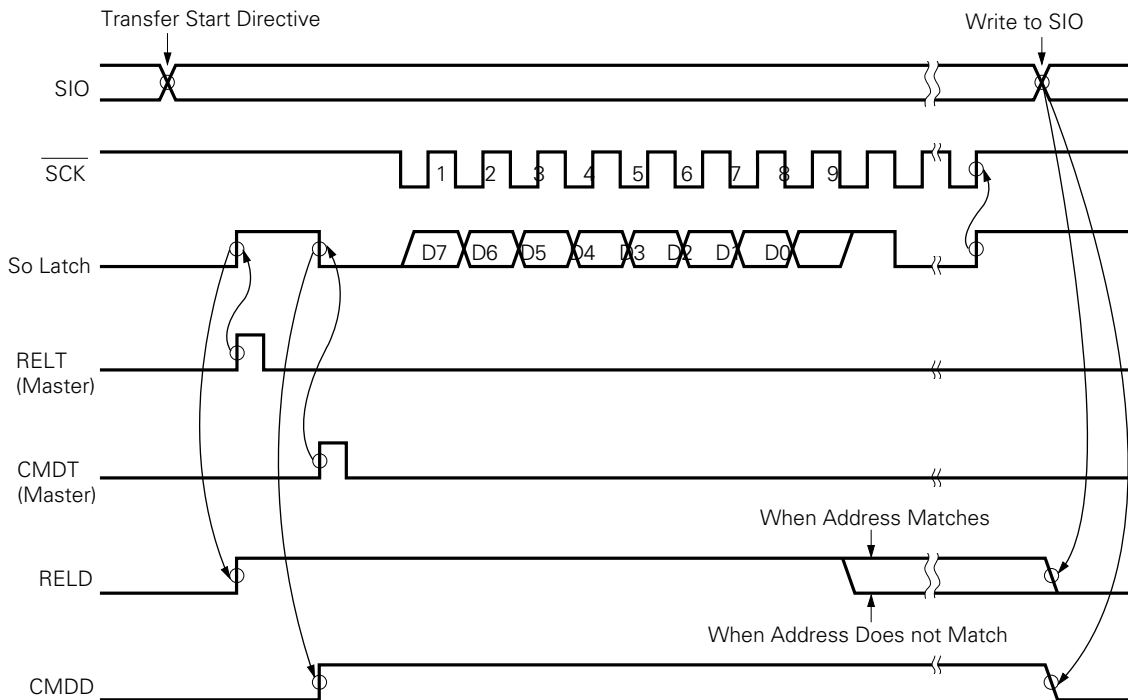
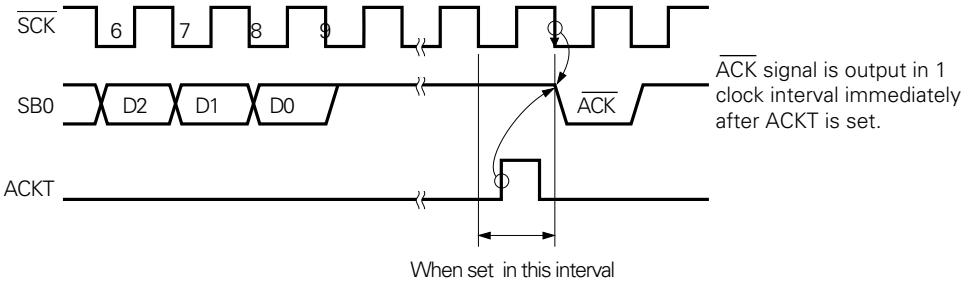


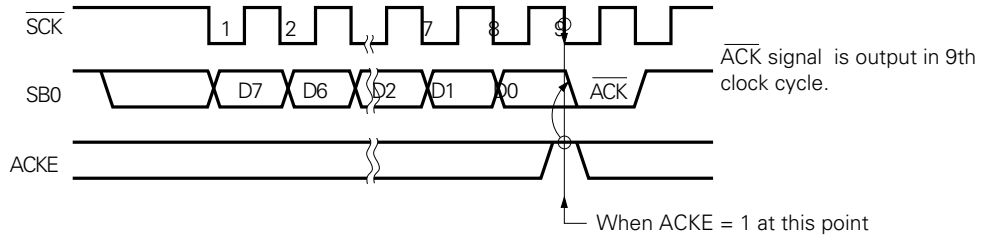
Fig. 5-44 ACKT Operation



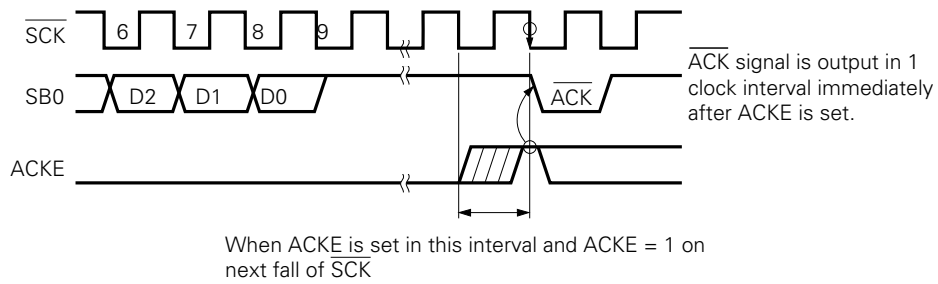
Note ACKT must not be set before the end of a transfer.

Fig. 5-45 ACKE Operation

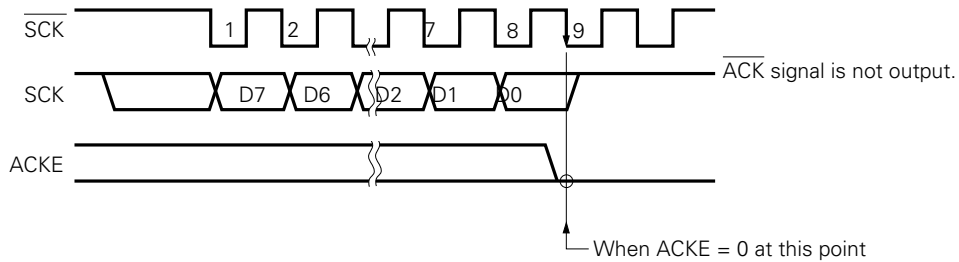
(a) When ACKE = 1 on completion of transfer



(b) When ACKE is set after completion of transfer



(c) When ACKE = 0 on completion of transfer



(d) When ACKE = 1 interval is short

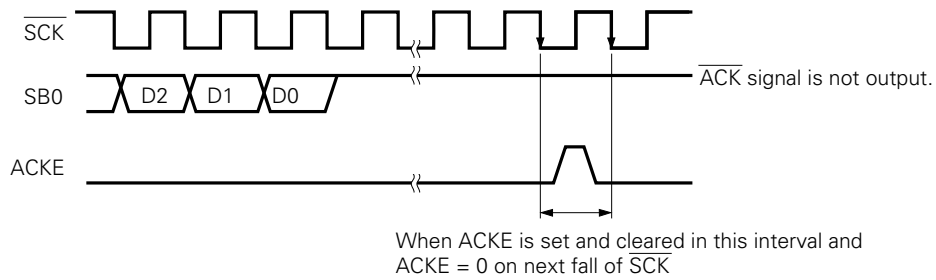
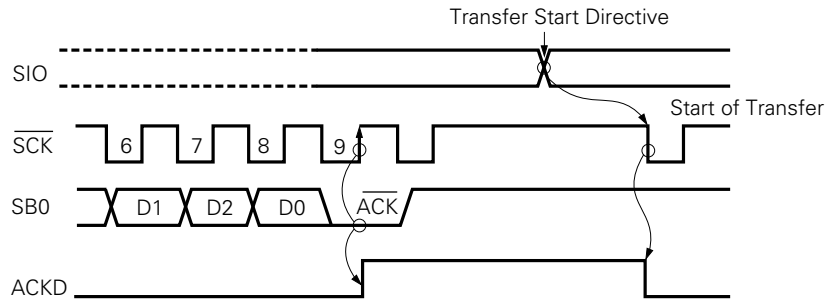
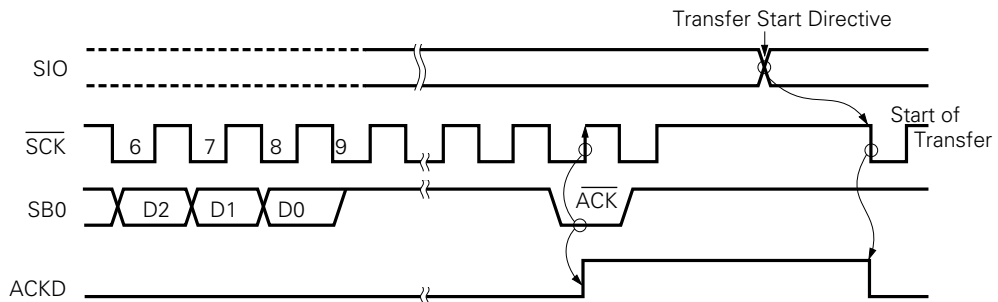


Fig. 5-46 ACKD Operation

(a) When $\overline{\text{ACK}}$ signal is output in 9th $\overline{\text{SCK}}$ clock interval



(b) When $\overline{\text{ACK}}$ signal is output after 9th $\overline{\text{SCK}}$ clock interval



(c) Clearing timing when transfer start directive is given during busy state

★

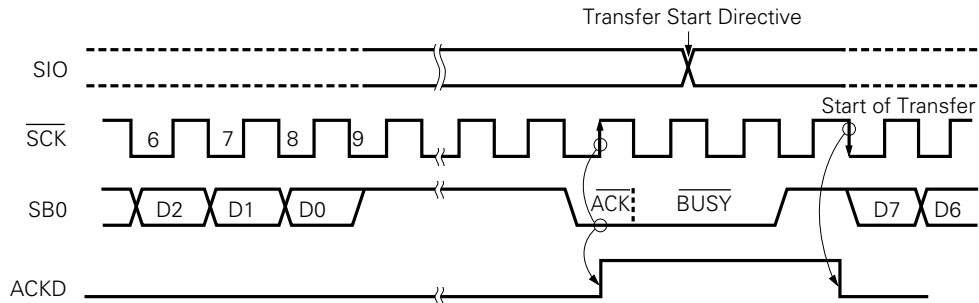


Fig. 5-47 BSYE Operation

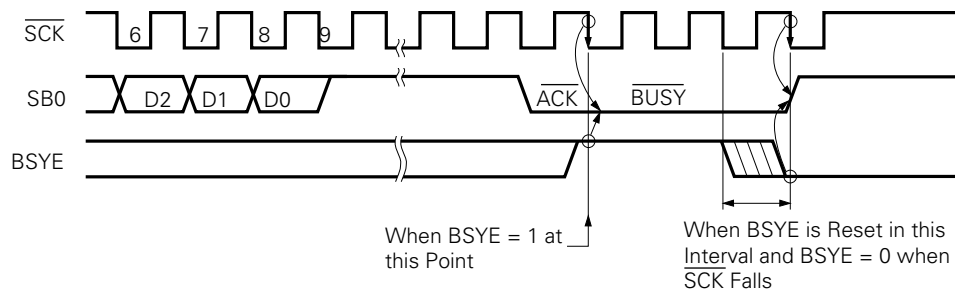


Table 5-8 Signals in SBI Mode (1/2)

Signal Name	Output Device	Definition	Timing Chart	Output Condition	Effect on Flag	Meaning of Signal
Bus release signal (REL)	Master	SB0 rising edge when $\overline{SCK} = 1$		• RELT set	• RELD set • CMDD Cleared	Outputs next CMD signal and indicates send data is address.
Command signal (CMD)	Master	SB0 falling edge when $\overline{SCK} = 1$		• CMDT set	• CMDD set	i) After REL signal output send data is address. ii) send data with no REL signal output is command.
Acknowledge signal (\overline{ACK})	Master/slave	Low-level signal output to SB0 in 1 \overline{SCK} clock interval after serial receive completion.		① ACKE = 1 ② ACKE set	• ACKD set	Receive completion
Busy signal (\overline{BSY})	Slave	Low-level signal output to SB0 after Acknowledge signal.		• BSYE = 1	—	Serial transmission/reception disabled because processing is in progress.
Ready signal (READY)	Slave	High-level signal output to SB0 before start or after completion of serial transfer.		① BSYE = 0 ② Execution of instruction to write data to SIO (transfer start directive)	—	Serial transmission/reception enabled

Table 5-8 Signals in SBI Mode (2/2)

Signal Name	Output Device	Definition	Timing Chart	Output Condition	Effect on Flag	Meaning of Signal
Serial Clock (SCK)	Master	Synchronization clock for output of address/command/data, ACK signal, Synchronous BUSY signal, etc. Address/command/ data is transferred in first 8 cycles.		Execution of instruction to write to SIO when CSIE = 1 (serial transfer start directive) *2	IRQCSI set (rise of 9th SCK clock cycle) *1	Timing of signal output to serial data bus
Address (A7 to A0)	Master	8-bit data transferred in synchronization with SCK after REL signal and CMD signal output.				Address value of slave device on serial bus
Command (C7 to C0)	Master	8-bit data transferred in synchronization with SCK after CMD signal only is output without output of REL signal.				Directive, message, etc., to slave device.
Data (D7 to D0)	Master/slave	8-bit data transferred in synchronization with SCK with no output of either REL signal or CMD signal.				Data to be processed by slave or master device

- * 1. When WUP = 0, IRQCSI is always set on the 9th rise of SCK.
When WUP = 1, IRQCSI is set on the 9th rise of SCK only with an address is receive and that address matches the value of the slave address register (SVA).
2. In data transmission/reception, when in the BUSY state, the transfer starts after transition to the READY state.

(6) Pin configuration

The configuration of the serial clock pin ($\overline{\text{SCK}}$) and the serial data bus pin SB0 is as shown below.

(a) $\overline{\text{SCK}}$ Pin for input/output of serial clock

① Master CMOS, push-pull output

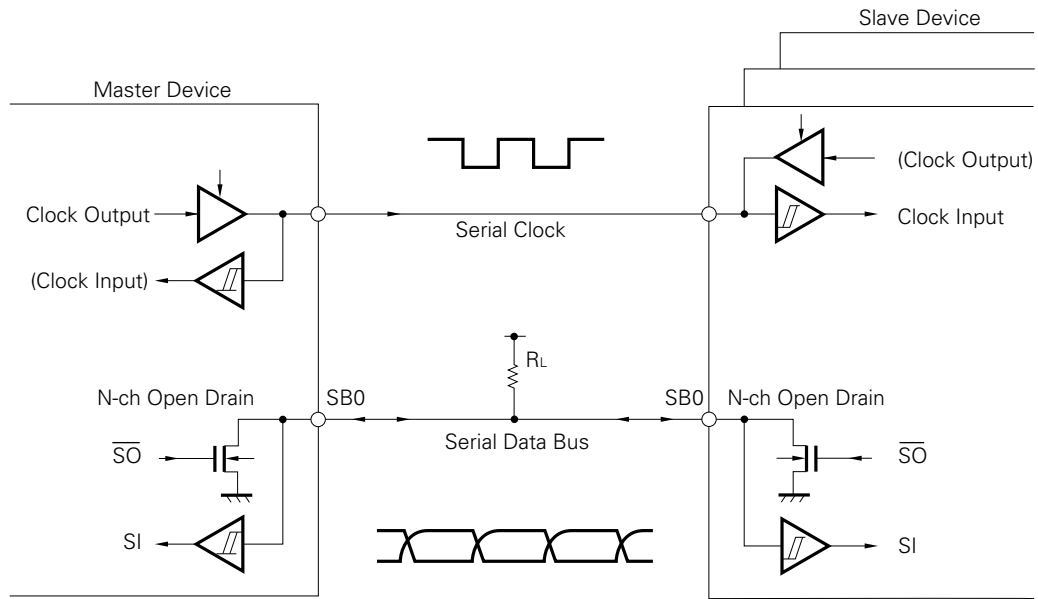
② Slave Schmitt input

(b) SB0 Serial data input/output dual- function pin

For both master and slave, output is N-ch open-drain, input is Schmitt input.

Since the serial data bus line output is N-ch open-drain, an external pull-up resistor is necessary.

Fig. 5-48 Pin Configuration Diagram



Note Since the N-ch open drain must be turned off during data reception, FFH should be written to SIO beforehand. It can always be turned off during transmission. However, when the wake-up function specification bit (WUP) is 1, the N-ch transistor is always off, and therefore FFH need not be written to SIO prior to reception.

(7) Address match detection method

In the SBI mode, master address communication is used to select a specific slave and start communication.

Address match detection is performed by hardware. A slave address register (SVA) is provided, and IRQCSI is set only when the address sent from the master and the value set is SVA match in the wake-up state (WUP = 1).

Note 1. Detection of the slave selected/nonselected state is performed by detection of a match with a slave address received after bus release (when RELD = 1).

An address match interrupt (IRQCSI) generated when WUP = 1 is normally used for this match detection. Therefore, detection of selection/nonselection by the slave address should be performed with WUP = 1.

2.

For selection/nonselection detection without using an interrupt when WUP = 0, the address detection method is not used: Instead, detection should be performed by transmission/reception of commands set beforehand by the program.

(8) Error detection

In the SBI mode, since the status of the serial bus SB0 pin during transmission is also written into the SIO shift register of the transmitting device, transmission errors can be detected in the following ways:

(a) Comparison of pre-transmission and post-transmission SIO data

In this case, a transmission error is judged to have been generated if the two data items are different.

(b) Use of slave address register (SVA)

Transmission is performed after also setting the send data in the SVA register. After transmission the COI bit of the serial operating mode register (CSIM) (the match signal from the address comparator) is tested: "1" indicates normal transmission, and "0", a transmission error.

(9) Communication operation

With the SBI, the master normally selects the slave device to be communicated with from among the multiple connected devices by outputting an address onto the serial bus.

After the target communication device has been determined, commands and data are exchanged between the master device and slave device, thus implementing serial communication.

Data communication timing charts are shown in Figs. 5-49 through 5-52.

In the SBI mode, shift register shift operations are performed in synchronization with the fall of the serial clock (\overline{SCK}), and send data is latched in the SO latch and is output MSB-first from the SB0/P02 or P03 pin. Receive data input to the SB0 pin is latched in the shift register on the rise of \overline{SCK} .

Fig. 5-49 Address Transmission from Master Device to Slave Device (WUP = 1)

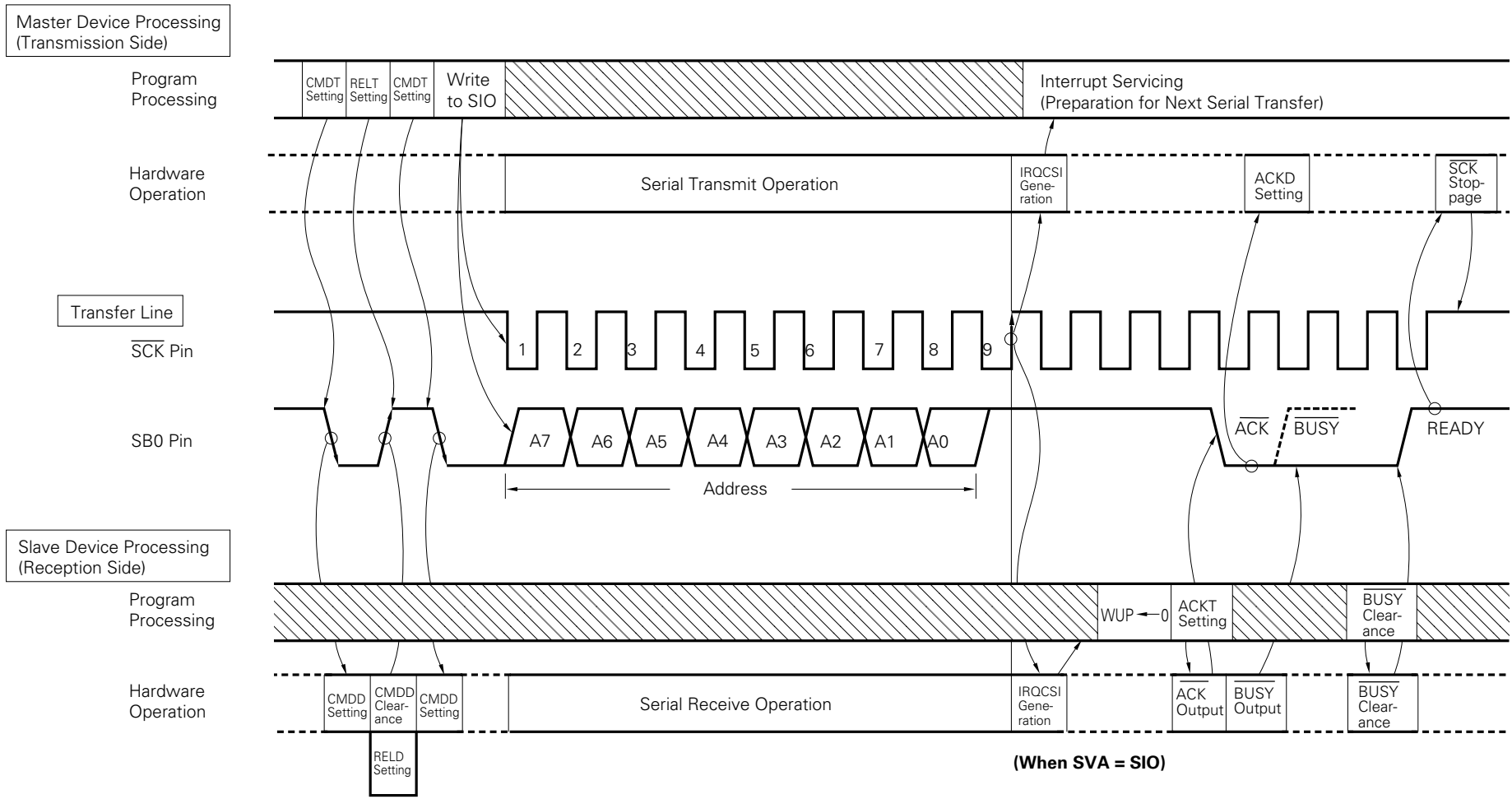


Fig. 5-50 Command Transmission from Master Device to Slave Device

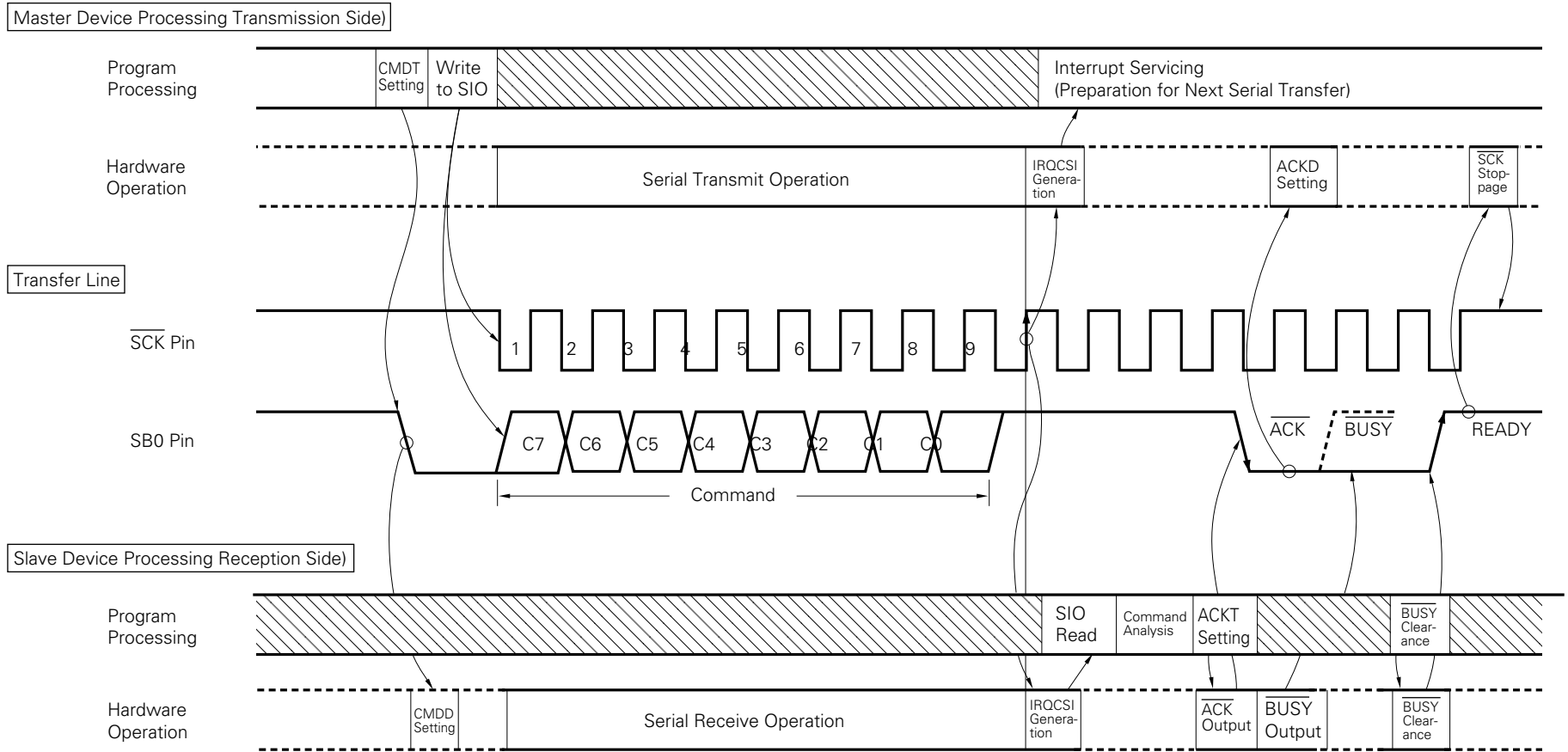


Fig. 5-51 Data Transmission from Master Device to Slave Device

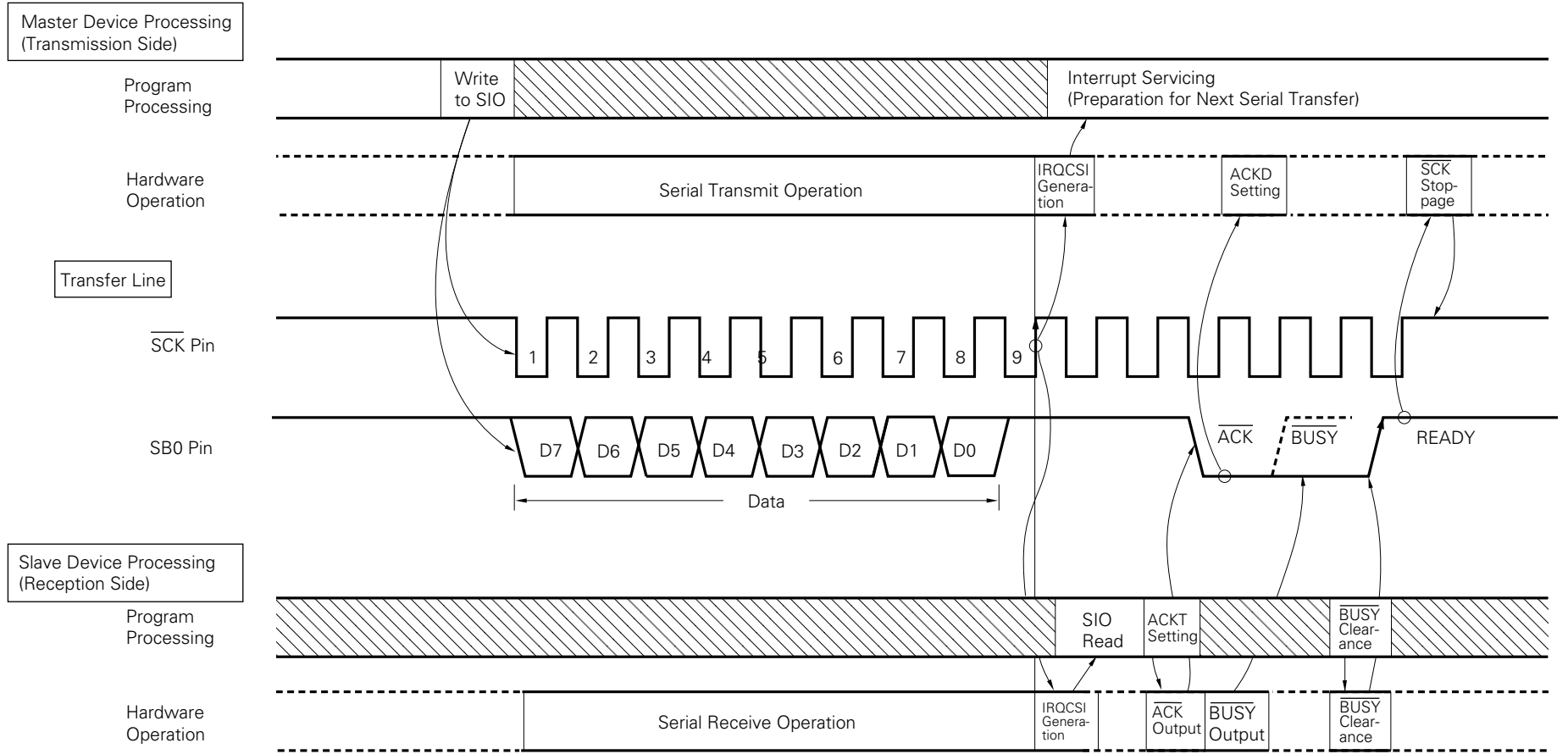
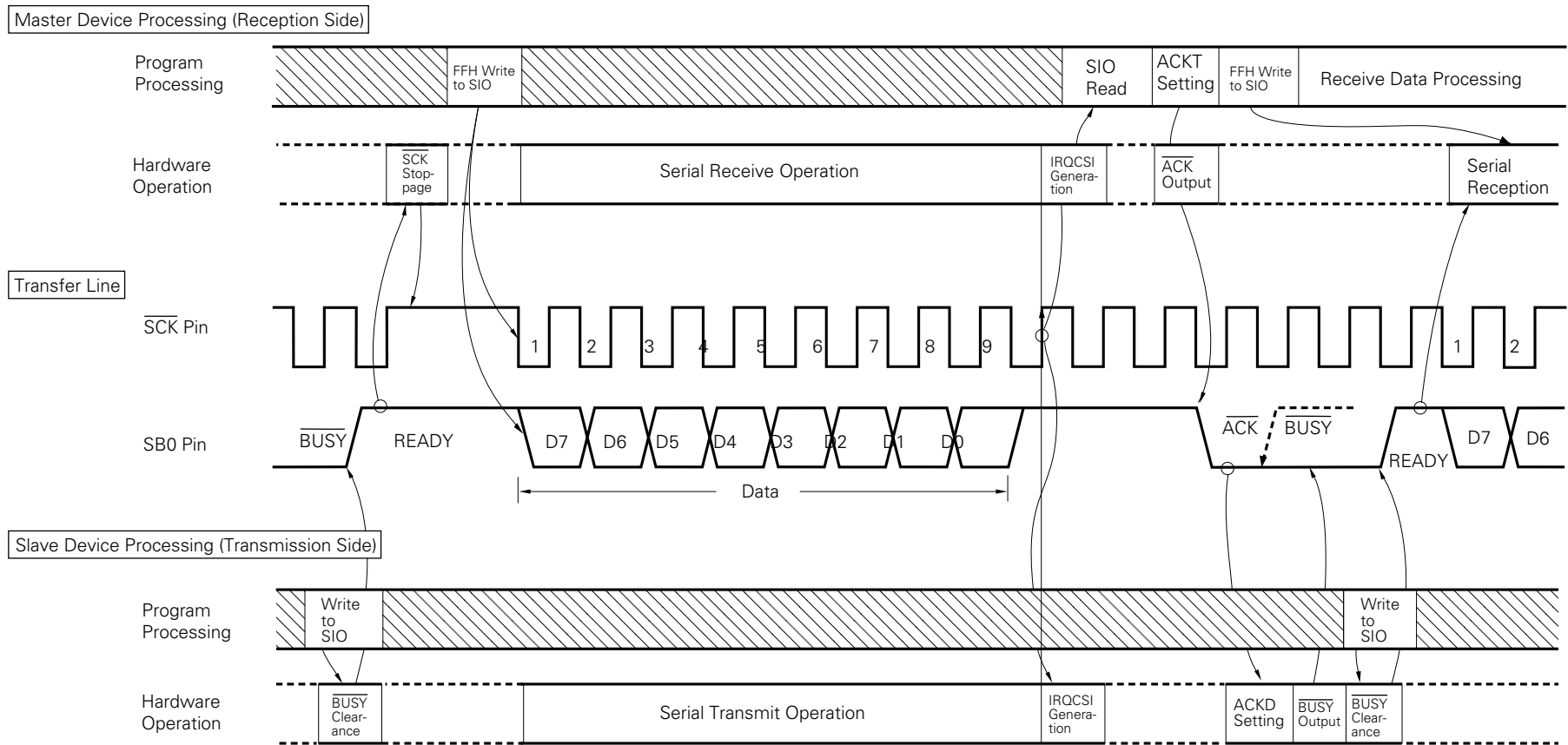


Fig. 5-52 Data Transmission from Slave Device to Master Device ★



(10) Start of transfer

When the following two conditions are met a serial transfer is started by setting transfer data in the shift register (SIO).

- The serial interface operatio enable/disable bit (CSIE) = 1.
- After an 8-bit serial transfer, the internal serial clock is stopped or $\overline{\text{SCK}}$ is high.

- Note**
1. The transfer will not be started if CSIE is set to "1" after data is written into the shift register.
 2. Since the N-ch transistor must be turned off during data reception, FFH should be written to SIO beforehand.
However, when the wake-up function specification bit (WUP) is 1, the N-ch transistor is always off, and therefore FFH need not be written to SIO prior to reception.
 3. If data is written to SIO when the slave is in the busy state, that data is not lost.
The transfer starts when the busy state is released and the SB0 input becomes high (ready state).

When an 8-bit transfer ends, the serial transfer stops automatically and the IRQCSI interrupt request flag is set.

Example In the following example the data in the RAM specified by the HL register is transferred to SIO, and at the same time the SIO data is fetched into the accumulator and the serial transfer is started.

```
MOV  XA, @HL ; Fetch send data from RAM
XCH  XA, SIO ; Exchange send data with receive data and start transfer
```

(11) Points to note concerning SBI mode

- (a) Detection of the slave selected/nonselected state is performed by detection of a match with a slave address received after bus release (when RELD = 1).
An address match interrupt (IRQCSI) generated when WUP = 1 is normally used for this match detection. Therefore, detection of selection/nonselection by the slave address should be performed with WUP = 1.
- (b) For selection/nonselection detection without using an interrupt when WUP = 0, the address detection method is not used: Instead, detection should be performed by transmission/reception of commands set beforehand by the program.
- (c) When WUP is set to 1 during $\overline{\text{BUSY}}$ signal output, $\overline{\text{BUSY}}$ is not released. With the SBI, the $\overline{\text{BUSY}}$ signal is output following a $\overline{\text{BUSY}}$ release directive until the next fall of the serial clock ($\overline{\text{SCK}}$). When WUP is set to 1, a check must be performed to ensure that SB0 has been driven high after $\overline{\text{BUSY}}$ is released before setting WUP to 1.

(12) SBI mode application

This section presents examples of applications in which serial data communication is performed in SBI mode. In these application examples, the μ PD75402A is operated as a slave CPU on the serial bus.

Also, the master can be changed by a command.

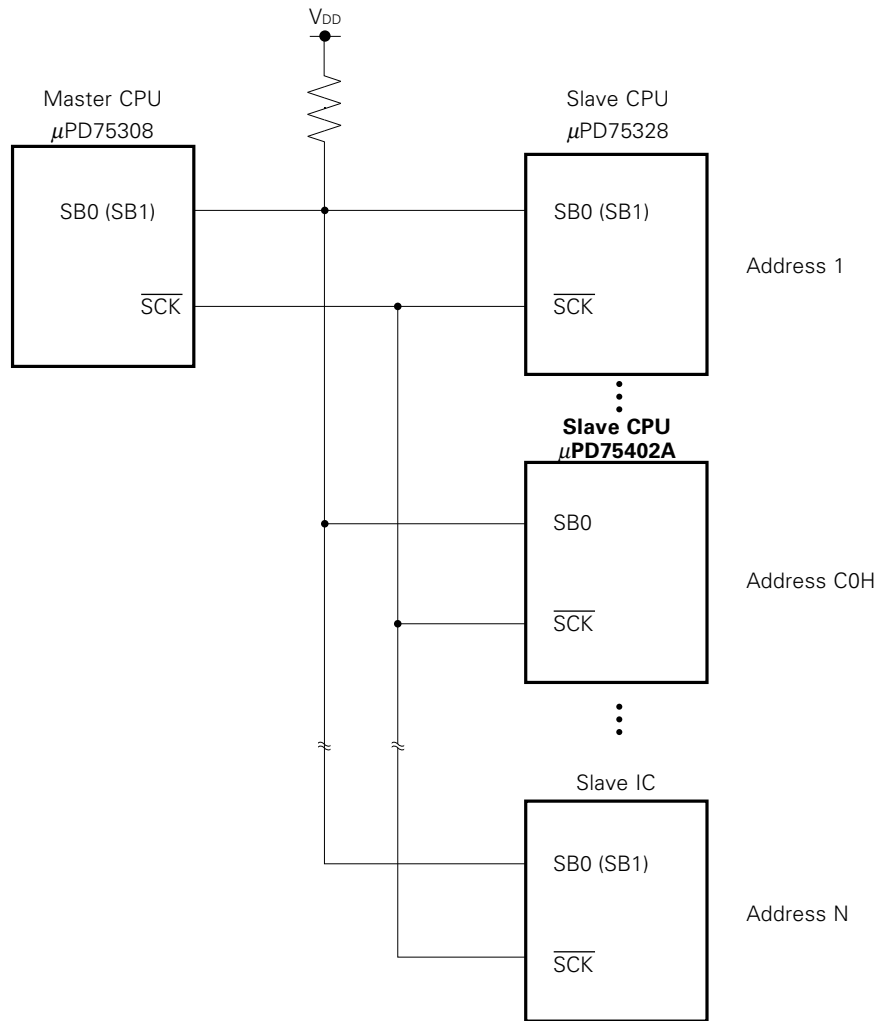
(a) Serial bus configuration

In the serial bus configuration in the application examples given here, the μ PD75402A is connected to the bus line as one of the devices on the serial bus.

The μ PD75402A uses two pins: The serial data bus SB0 (P02/SO), and the serial clock $\overline{\text{SCK}}$ (P01).

An example of the serial bus configuration is shown in Fig. 5-53. Only addresses C0H through C7H can be allocated to the μ PD75402A.

Fig. 5-53 Example of Serial Bus Configuration



(b) Description of commands**(i) Command types**

The following command types are used in these application examples.

- ① READ command : Performs data transfer from slave to master.
- ② WRITE command : Performs data transfer from master to slave.
- ③ END command : Notifies slave of completion of WRITE command.
- ④ STOP command : Notifies slave of suspension of WRITE command.
- ⑤ STATUS command : Reads slave-side status.
- ⑥ RESET command : Makes currently selected slave non-selected.
- ⑦ CHGMST command : Passes mastership to slave side.

(ii) Communication procedure

The procedure for communication between master and slave is as follows.

- ① The master starts the communication by transmitting the address of the slave to be communicated with and selecting the slave (chip selection).

When the slave receives the address, it communicates with the master by returning $\overline{\text{ACK}}$. (The slave changes from non-selected to selected status.)

- ② Communication is performed between the slave selected by the processing in ① and the master by the transfer of commands and data.

However, as command and data transfers are one-to-one master- slave transfers, other slaves must be in the non-selected status.

- ③ Communication is ended by the slave changing to non-selected status. The slave changes to non-selected status in the following cases:
 - When a RESET command is sent from the master, the selected slave changes to non-selected status.
 - When the master is changed by the CHGMST command, the device which is changed from master to slave changes to non-selected status.

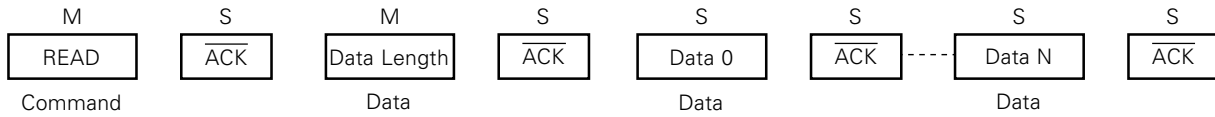
(iii) **Command formats**

The transfer format of each command is shown below.

① **READ command**

This command performs a read from the slave. The read data length is variable between 1 and 256 bytes, and is specified as a parameter by the master. If 00H is specified as the data length, this is interpreted as a 256-byte data transfer specification.

Fig. 5-54 READ Command Transfer Format



Remarks M : Output by master
S : Output by slave

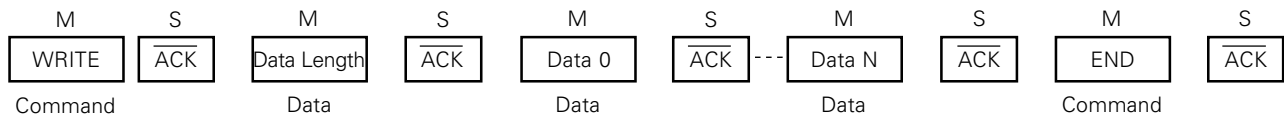
After the slave receives the data length, if the transmissible data is equal to or greater than that data length, the slave returns $\overline{\text{ACK}}$. If the data is insufficient, $\overline{\text{ACK}}$ is not returned and an error is generated.

When a data transfer is performed, the slave compares the SIO contents before and after the transfer to check whether the data was correctly output onto the bus. If the SIO contents before and after the transfer are different, $\overline{\text{ACK}}$ is not returned and an error is generated.

② **WRITE, END and STOP commands**

This command performs a data write to the slave. The write data length is variable between 1 and 256 bytes, and is specified as a parameter by the master. If 00H is specified as the data length, this is interpreted as a 256-byte data transfer specification.

Fig. 5-55 WRITE & END Command Transfer Format



Remarks M : Output by master
S : Output by slave

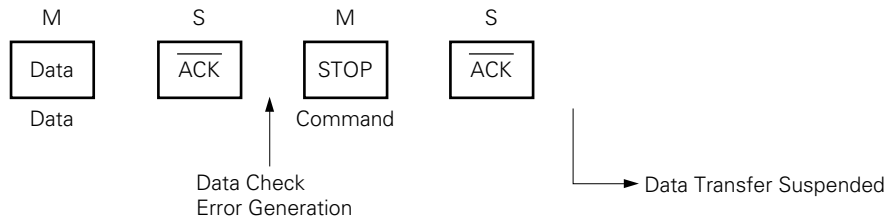
After the slave receives the data length, if the area for storing the receive data is at least as large as that data length, the slave returns $\overline{\text{ACK}}$. If the data storage area is too small, $\overline{\text{ACK}}$ is not returned and an error is generated.

When all the data has been transferred, the master sends an END command. This command notifies the slave that all the data has been correctly transferred.

The slave may also receive an END command before the reception of all the data. In this case, the data up to the reception of the END command is valid.

When data is transmitted, the master compares the SIO contents before and after the transmission to check whether the data was correctly output onto the bus. If the SIO contents before and after the transmission are different, the master suspends data transmission by sending a STOP command.

Fig. 5-56 STOP Command Transfer Format



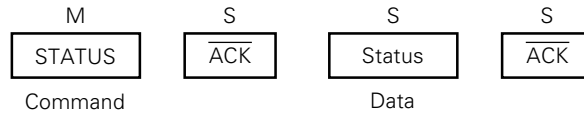
Remarks M : Output by master
S : Output by slave

When the slave receives a STOP command, the byte of data received immediately before the command is invalid.

③ **STATUS command**

This command is used to read the status of the currently selected slave.

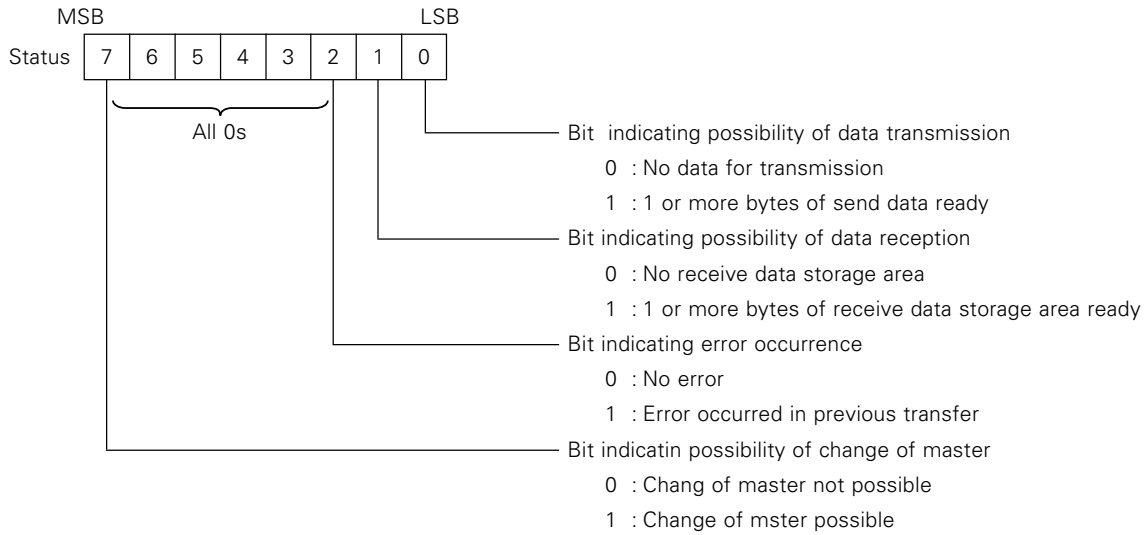
Fig. 5-57 STATUS Command Transfer Format



Remarks M : Output by master
S : Output by slave

The format of the status byte returned by the slave is shown below.

Fig. 5-58 STATUS Command Status Format

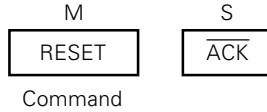


When the slave sends status data, it compares the SIO contents before and after transmission and if they do not match an error is generated and no $\overline{\text{ACK}}$ is returned.

④ **RESET command**

This command is used to change the currently selected slave to non-selected status. All slaves can be placed in non-selected status by sending the RESET command.

Fig. 5-59 RESET Command Transfer Format

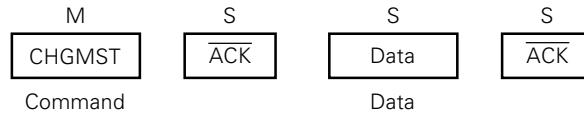


Remarks M : Output by master
S : Output by slave

⑤ **CHGMST command**

This command passes mastership to the currently selected slave.

Fig. 5-60 CHGMST Command Transfer Format



Remarks M : Output by master
S : Output by slave

When a slave receives the CHGMST command, it determines whether it can assume mastership and returns data to the master. This data is as follows:

- 0FFH : Change of master possible
- 00H : Change of master not possible

When the slave sends this data, it compares the SIO contents before and after transmission and if they do not match an error is generated and no $\overline{\text{ACK}}$ is returned.

If there is no error after the 0FFH data has been transmitted, the master operates as a slave and the slave operates as the master from that point on.

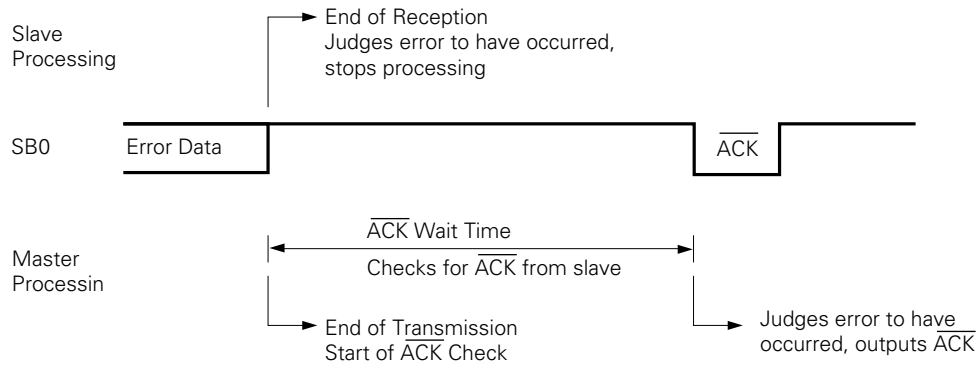
(iv) Error occurrence

Operation in the event of an error in communication is described below.

A slave indicates the occurrence of an error by failing to return $\overline{\text{ACK}}$ to the master. When an error occurs, the status bit indicating the occurrence of an error is set and all command processing being executed is canceled.

After sending or receiving a byte, the master checks for $\overline{\text{ACK}}$ from the slave. If $\overline{\text{ACK}}$ is not returned by the slave within a certain time after the end of transmission or reception, an error is judged to have occurred and the master outputs an $\overline{\text{ACK}}$ signal (as a dummy).

Fig. 5-61 Master and Slave Operations after an Error



Errors are generated in the following circumstances:

• **Errors generated on the slave side**

- ① If the command transfer format is wrong.
- ② If an undefined command is received.
- ③ If the transferred data length is insufficient in a READ command.
- ④ If the data storage area is too small in a WRITE command.
- ⑤ If the data changes in a READ, STATUS or CHGMST command data transmission.

$\overline{\text{ACK}}$ is not returned if any of the above cases.

• **Errors generated on the master side**

If the data changes in a WRITE command data transmission, a STOP command is sent to the slave.

CHAPTER 6. INTERRUPT FUNCTIONS

On the μ PD75402A there are 3 vectored interrupt sources and one testable input, enabling a wide variety of applications to be handled.

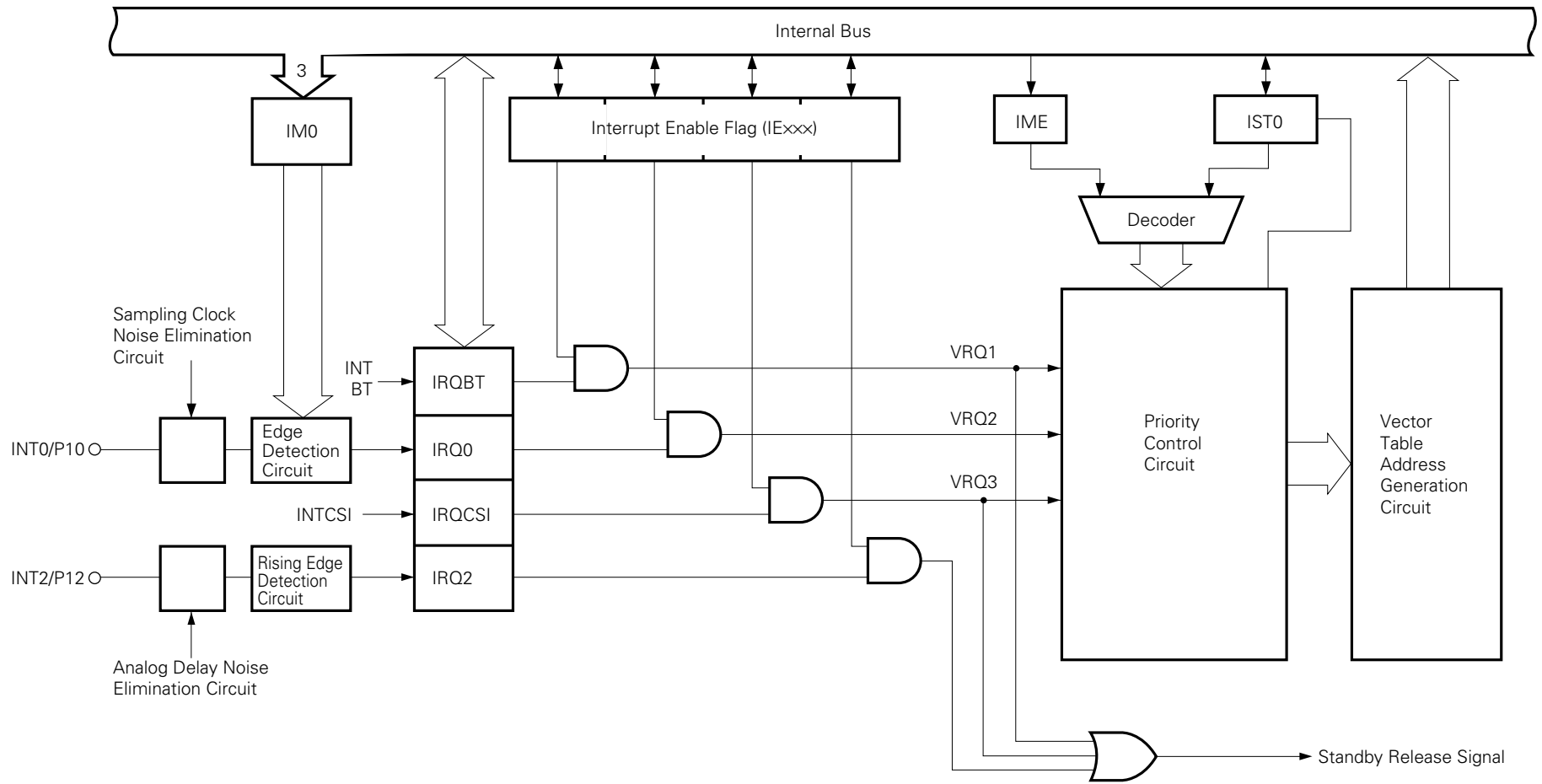
Moreover, the μ PD75402A's interrupt control circuit has the following special features, making possible extremely fast interrupt servicing.

- (a) Acknowledgment enabling/disabling is possible by means of the interrupt master enable flag (IME) and the interrupt enable flag (IE_{xxx}).
- (b) Any desired interrupt service start address can be set using the vector table (for rapid starting of the actual interrupt service program).
- (c) Interrupt request flag (IRQ_{xxx}) can be tested and cleared (allowing checking of interrupt generation by software).
- (d) Standby mode (HALT) can be released by an interrupt request (release source is selectable from other than INT0 by means of interrupt enable flag).

6.1 INTERRUPT CONTROL CIRCUIT CONFIGURATION

The interrupt control circuit is configured as shown in Fig. 6-1, with each hardware item mapped onto data memory space.

Fig. 6-1 Interrupt Control Circuit Block Diagram



6.2 INTERRUPT SOURCE TYPES AND VECTOR TABLE

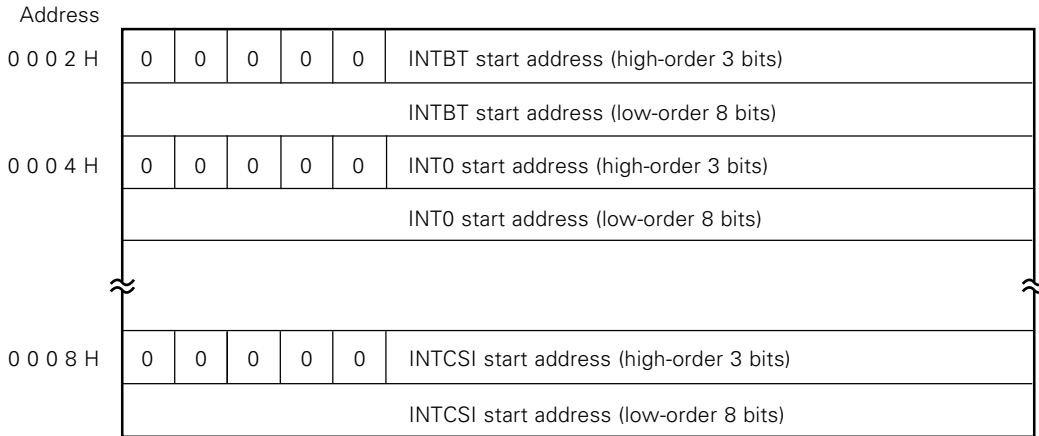
The μ PD75402A's interrupt source types and interrupt vector table are shown in Table 6-1 and Fig. 6-2.

Table 6-1 Interrupt Request Source Types

Interrupt Request Generation Source	Internal/ External	Interrupt Priority*1	Vectored Interrupt Request Signal (Vector Table Address)
INTBT (Basic time interval signal from basic interval timer)	Internal	1	VRQ1(0002H)
INT0 (INT0 pin input specified edge detection)	External	2	VRQ2(0004H)
INTCSI (Serial data transfer termination signal)	Internal	3	VRQ3(0008H)
INT2*2 (INT2 pin input rising edge detection)	External	Testable input signal (sets) IRQ2 flag)	

- * 1. The interrupt priority is the order of precedence when multiple interrupt requests occur simultaneously.
 2. A test source. This is affected by an interrupt enable flag in the same way as an interrupt source, but no vectored interrupt is generated.

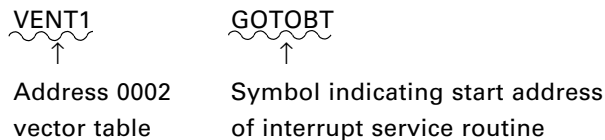
Fig. 6-2 Interrupt Vector Table



The interrupt priority shown in the table shows the order in which interrupts are executed when multiple interrupt requests are generated simultaneously or when multiple interrupt requests are pending.

The vector table contains the interrupt service routine start addresses. Vector table setting is performed by the VENTn assembler pseudoinstruction.

Example INTBT vector table setting.



Note The vector table address specified by VENTn (n = 1, 2 and 4) is address 2n.

Example INTBT and INT0 vector table setting.

VENT1 GOTBT
 VENT2 GOTO0

6.3 INTERRUPT CONTROL CIRCUIT HARDWARE

(1) Interrupt request flag & interrupt enable flag

There are four interrupt request flags (IRQ_{xxx}) corresponding to the interrupt sources (interrupt: 3, test: 1) as follows.

- INT0 interrupt request flag (IRQ0)
- INT2 interrupt request flag (IRQ2)
- BT interrupt request flag (IRQBT)
- Serial interface interrupt request flag (IRQCSI)

An interrupt request flag is set (1) by generation of an interrupt request and cleared (0) automatically by execution of an interrupt service.

There are four interrupt enable flags (IE_{xxx}) corresponding to the interrupt request flags as follows.

- INT0 interrupt enable flag (IE0)
- INT2 interrupt enable flag (IE2)
- BT interrupt enable flag (IEBT)
- Serial interface interrupt enable flag (IECSI)

An interrupt enable flag enables an interrupt when its contents are “1” and disables it when 0.

When an interrupt request flag is set and the interrupt enable flag permits an interrupt, a vectored interrupt request (VRQ_n) is generated. This signal is also used to release standby mode (HALT mode) (with the exception of VRQ2).

The interrupt request flags and interrupt enable flags are manipulated by bit-handling instructions and 4-bit memory handling instructions. In addition, the interrupt enable flags are manipulated by the EI IE_{xxx} instruction and the DI IE_{xxx} instruction. The SKTCLR is normally used for interrupt request flag testing.

```

Example EI          IE0          ; INT0 enabled
          DI          IEBT         ; INTBT disabled
          SKTCLR     IRQCSI        ; Skip and clear if IRQCSI is 1
    
```

When the interrupt request flag is set by an instruction, although no interrupt is generated, a vectored interrupt is executed in the same way as when an interrupt is generated.

With a RESET input, the interrupt request flags and interrupt enable flags are cleared (0) and all interrupts are disabled.

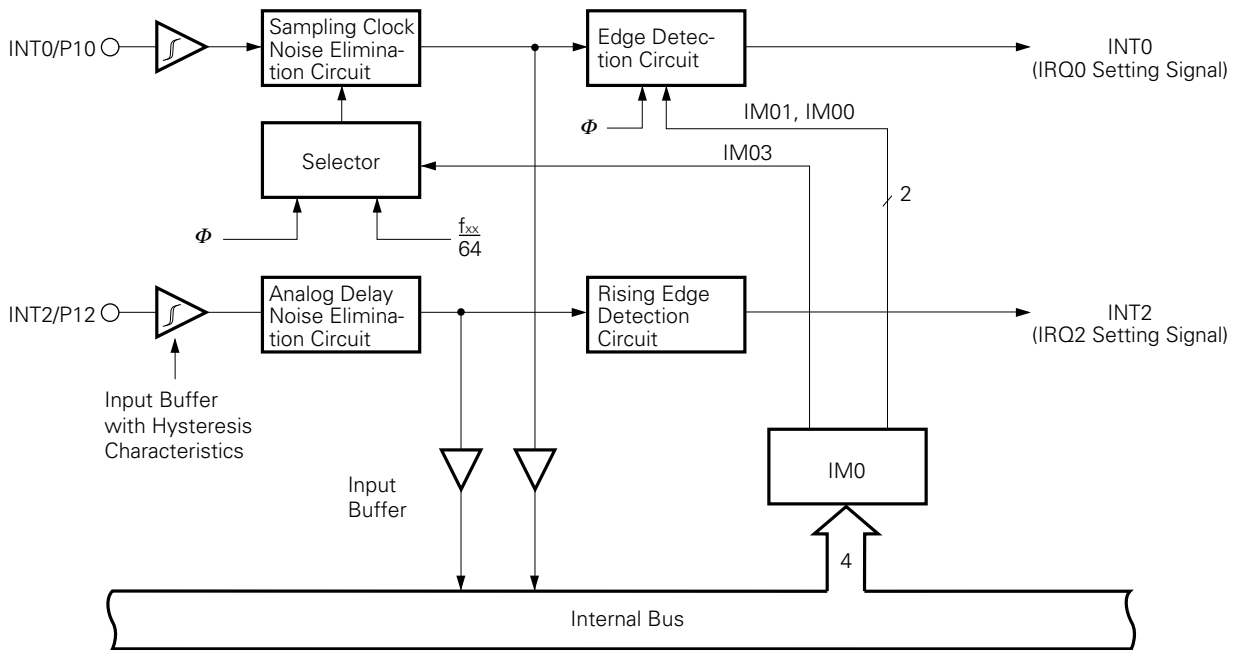
Table 6-2 Interrupt Request Flag Setting Signal

Interrupt Request Flag	Interrupt Request Flag Setting Signal	Interrupt Enable Flag
IRQBT	Set by basic time interval signal from basic interval timer.	IEBT
IRQ0	Set by INT0/P10 pin input signal edge detection. Detected edge is selected by INT0 mode register (IM0).	IE0
IRQCSI	Set by serial interface serial data transfer operation termination signal.	IECSI
IRQ2	Set by INT2/P12 pin input signal rising edge detection.	IE2

(2) External interrupt input pin hardware

The configuration of INT0 and INT2 is shown in Fig. 6-3.

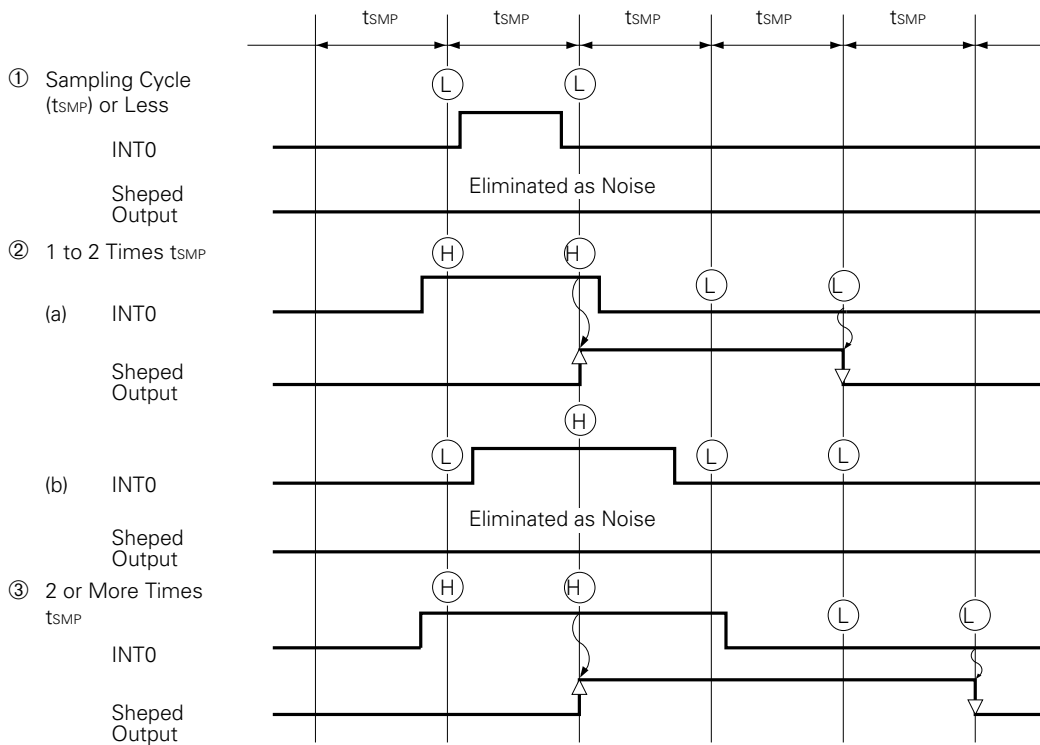
Fig. 6-3 Configuration of INT0 and INT2



INT0 functions as an external interrupt input on which sampling clock noise elimination and detected edge selection can be performed. The INT0 noise elimination circuit detects a change in level in 2 sampling clock pulses. Therefore, pulses narrower than the width of the 2 cycles ($2t_{CY}^*$ or $128/f_{XX}$) of sampling clock are eliminated as noise, and a pulse exceeding the width is properly acknowledged as an interrupt signal (see Fig. 6-4). One of 3 clocks can be selected as the timing clocks.

* Cycle time

Fig. 6-4 INT0 Noise Elimination Circuit Input/Output Timing



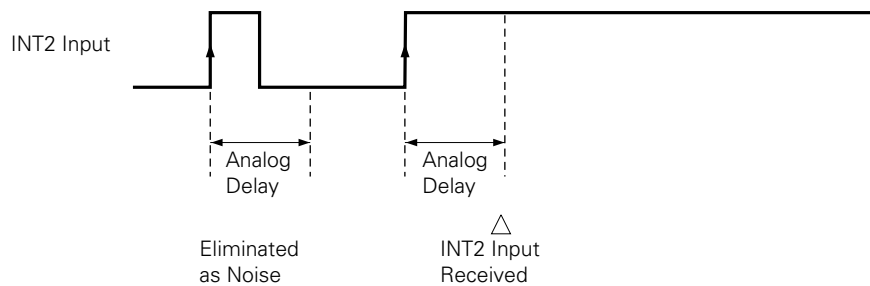
Remarks $t_{SMP} = t_{CY}$ or $64/f_{XX}$

Specification of the detected edge of the INT0 input and selection of the sampling clock is performed by the edge detection mode register (IM0).

As signals are also input via the noise elimination circuit when the INT0 pin inputs data as a port, the input data must be of sufficient width to avoid being eliminated as noise.

INT2 functions as an externally testable input which sets a testable flag on detection of a rising edge. Noise elimination by the sampling clock is not performed, but as there is a function for eliminating pulses which are narrower than the analog delay, a signal of adequate width must be input as in the case of INT0 (see Fig. 6-5).

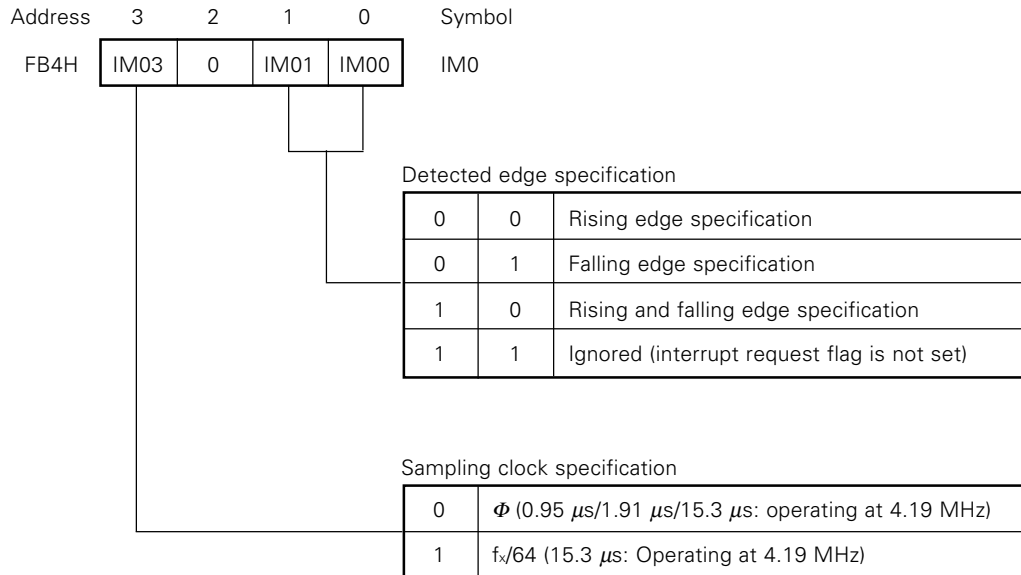
Fig. 6-5 INT2 Input Noise Elimination



The format of the edge detection mode register (IM0) which is used to select the detected edge is shown in Fig. 6-6. IM0 is set by 4-bit memory handling instructions.

On an $\overline{\text{RESET}}$ input, all bits of IM0 are cleared to 0 and the rising edge is specified for INT0.

Fig. 6-6 Edge Detection Mode Register Format



Note As the interrupt request flag may be set when the edge detection mode register is modified, the following procedure should be used: Disable interrupts and modify the mode register in advance, clear the interrupt request flag with the CLR1 instruction, and then enable interrupts again. Also, when $f_x/64$ is selected as the sampling clock by modifying IM0, the interrupt request flag must be cleared after the elapse of 16 machine cycles following the mode register modification.

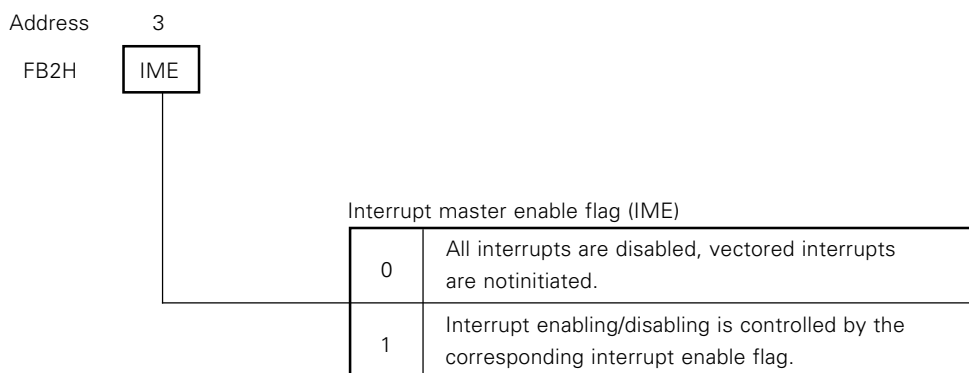
(3) Interrupt master enable flag (IME)

The interrupt master enable flag specifies acknowledgment enabled/disabled for all interrupts.

IME is manipulated by the EI/DI instructions.

With a $\overline{\text{RESET}}$ input, IME is cleared to 0 and acknowledgment of all interrupts is disabled.

Fig. 6-7 IME Format



(4) Interrupt status flag

The interrupt status flag (IST0) is the flag which shows the status of the processing currently being executed by the CPU, and is contained in the PSW.

The interrupt priority control circuit performs interrupt control according to the contents of this flag as shown in Table 6-3.

IST0 cannot be modified by 4-bit handling instructions or bit-handling instructions. IST0 is always set to 1 during interrupt servicing. Therefore, it is not possible to write 0 to IST in the interrupt service routine which would result in multiple interrupt.

After being saved to stack memory together with the rest of the PSW when an interrupt is acknowledged, IST0 is automatically set to 1. When an RETI instruction is executed, the original IST0 value (0) is restored.

A $\overline{\text{RESET}}$ input clears (0) the flag contents.

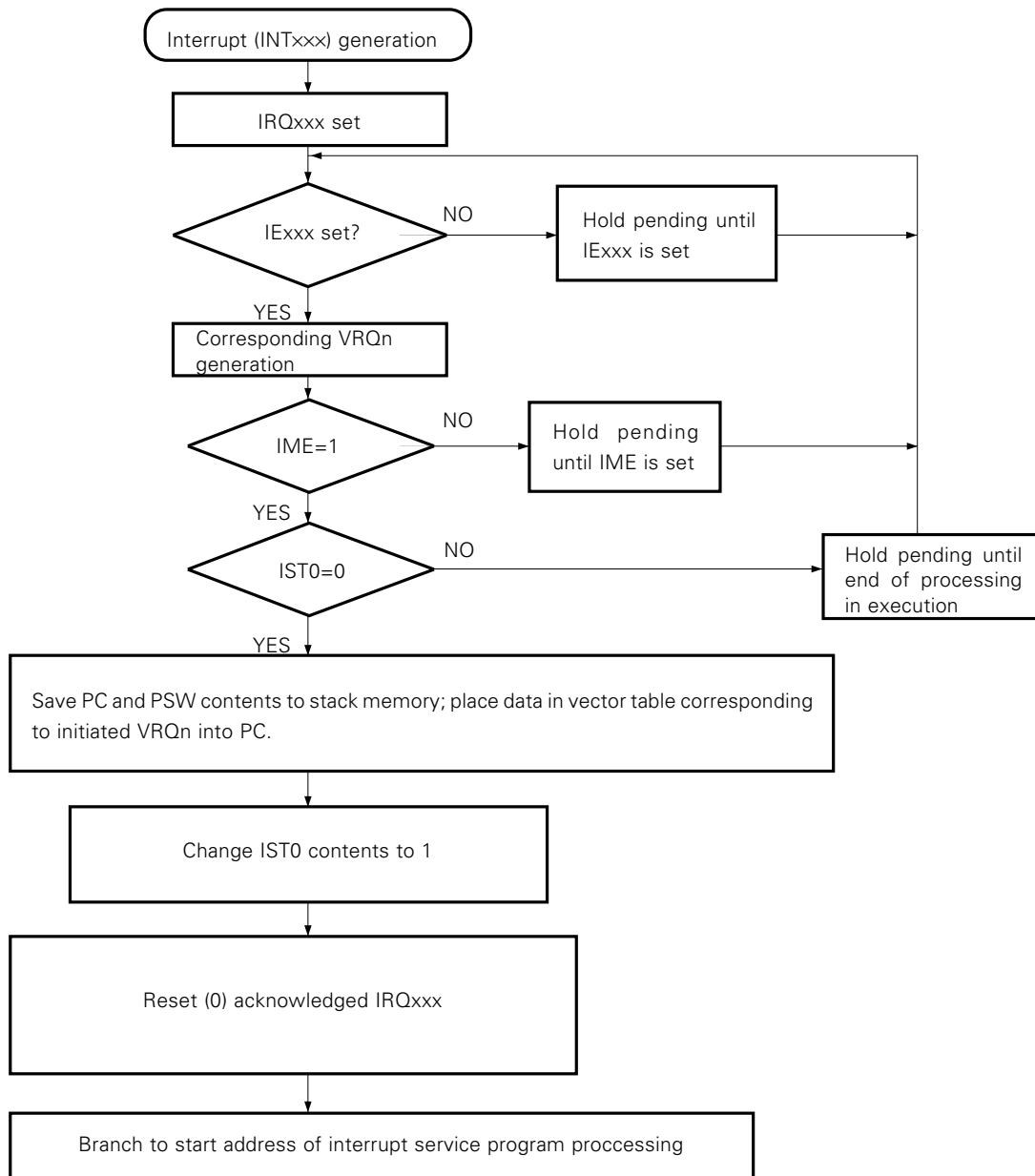
Table 6-3 IST0 Interrupt Servicing Status

IST0	Executing Processing Status	CPU Processing	Interrupt Requests of which Acknowledgment is Possible	After Interrupt Acknowledgment
				IST0
0	Status 0	Normal program processing in progress	Acknowledgment of all interrupts possible	1
1	Status 1	Interrupt servicing in progress	Acknowledgment of all interrupts disabled	–

6.4 INTERRUPT SEQUENCE

When an interrupt is generated, it is serviced by the procedure shown in Fig. 6-8.

Fig. 6-8 Interrupt Servicing Procedure

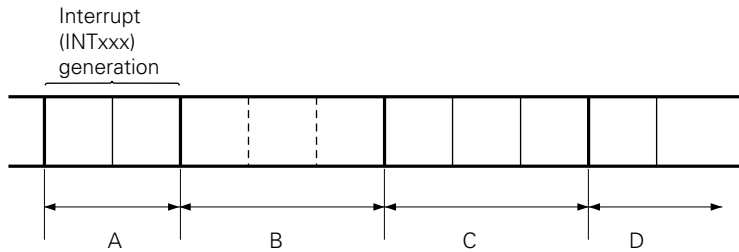


6.5 MACHINE CYCLES BEFORE INTERRUPT SERVICING

On the 75X, the machine cycles from the setting of the interrupt request flag (IRQn) until execution of the interrupt routine program are as shown below.

(1) When IRQn is set during execution of an interrupt control instruction

When IRQn is set during execution of an interrupt control instruction, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the next instruction.



- A : Setting of IRQn
- B : Execution of next instruction (between 1 and 3 machine cycles depending on instruction)
- C : Interrupt servicing (3 machine cycles)
- D : Execution of interrupt routine

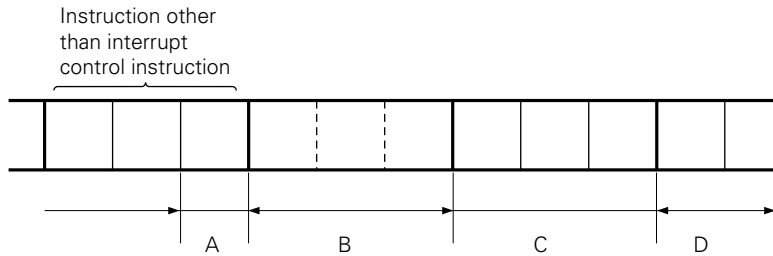
- Remarks
1. An interrupt control instruction is an instruction which manipulates interrupt-related hardware (data memory FBxH address). These instructions comprise the DI and EI instructions.
 2. The 3 machine cycles of interrupt servicing include the time for manipulation of the stack on acknowledgment of an interrupt, etc.

- Note
1. If there are a number of consecutive interrupt control instructions, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the last interrupt control instruction executed.
 2. When IRQn is set, or when the interrupt control instruction executed thereafter is a DI instruction, the interrupt request by which IRQn was set is held pending.

(2) When IRQn is set during execution of an instruction other than an interrupt control instruction

(a) When IRQn is set in the last machine cycle of the instruction being executed

In this case, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the instruction being executed.

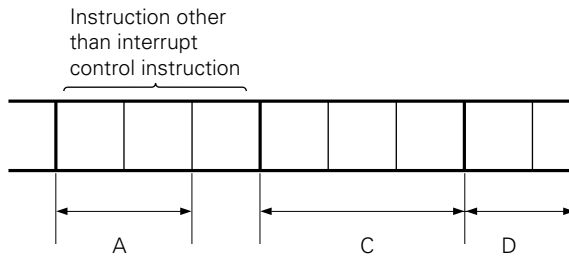


- A: Setting of IRQn
- B: Execution of next instruction (between 1 and 3 machine cycles depending on instruction)
- C: Interrupt servicing (3 machine cycles)
- D: Execution of interrupt routine

Note If the next instruction is an interrupt control instruction, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following execution of the instruction which follows the last interrupt control instruction executed. Also, if the interrupt control instruction executed after IRQn is set is a DI instruction, the interrupt request by which IRQn was set is held pending.

(b) When IRQn is set before the last machine cycle of the instruction being executed

In this case, the interrupt routine program is executed after 3 machine cycles of interrupt servicing have been performed following the instruction being executed.



- A: Setting of IRQn
- C: Interrupt servicing (3 machine cycles)
- D: Execution of interrupt routine

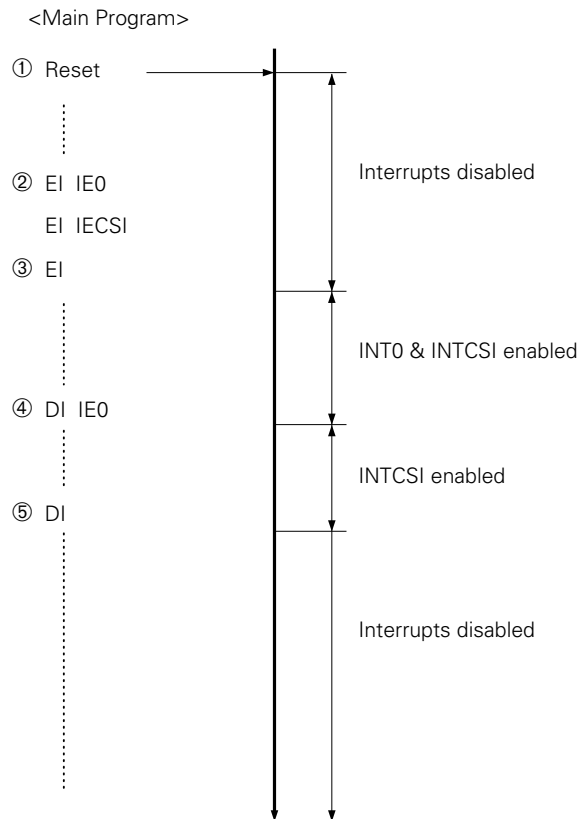
6.6 INTERRUPT APPLICATIONS

When the interrupt function is used, the following settings are first carried out in the main program.

- ① The interrupt enable flag corresponding to the interrupt to be used is set to "1" (EI IE_{xxx} instruction).
- ② If INT0 is used, the active edge is selected (IM0 setting).
- ③ The interrupt master enable flag (IME) is set to "1" (EI instruction).

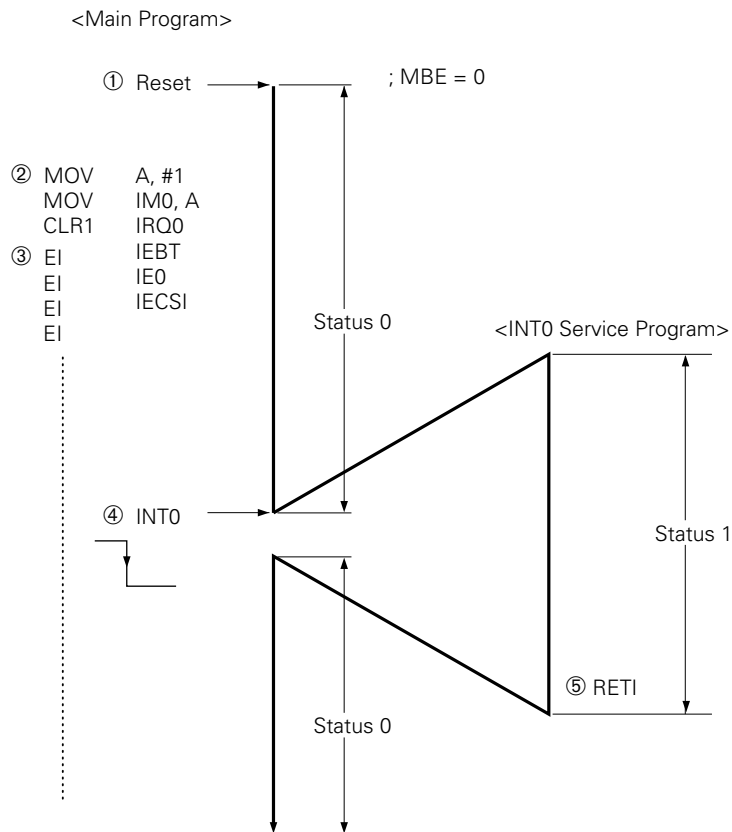
Return from the interrupt service program is by means of an RETI instruction.

(1) Interrupt enabling/disabling



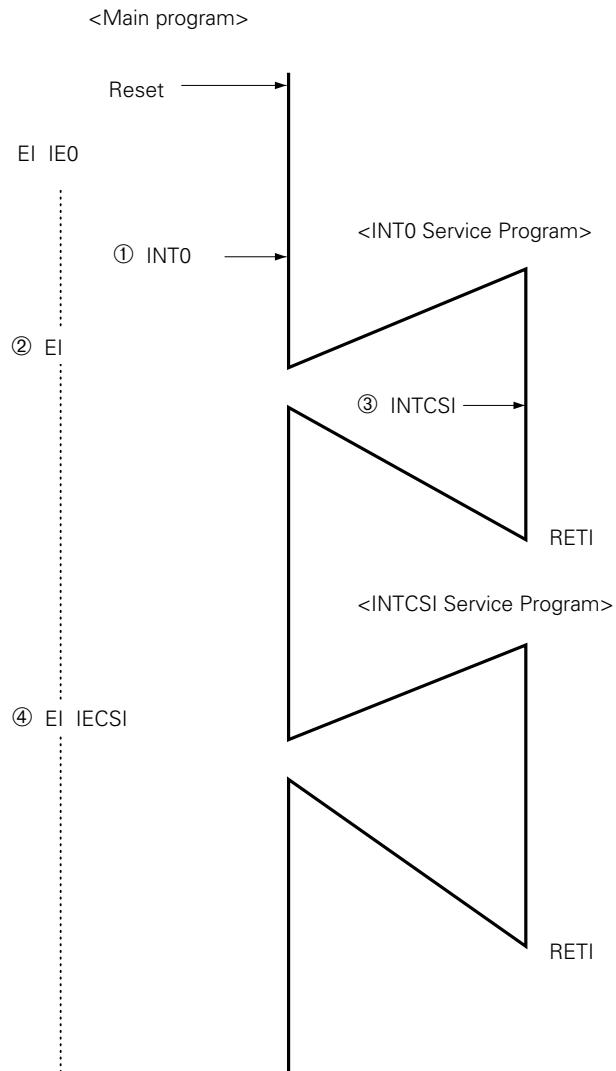
- ① All interrupts disabled by $\overline{\text{RESET}}$ input.
- ② Interrupt enable flag set by EI IE_{xxx} instruction.
At this stage, all interrupts are still disabled.
- ③ Interrupt master enable flag set by EI instruction.
At this stage, INT0 & INTCSI are enabled.
- ④ Interrupt enable flag cleared by DI IE_{xxx} instruction; INT0 disabled.
- ⑤ All interrupts disabled by DI instruction.

(2) Example using INTBT, INTO (falling edge active), and INTCSI



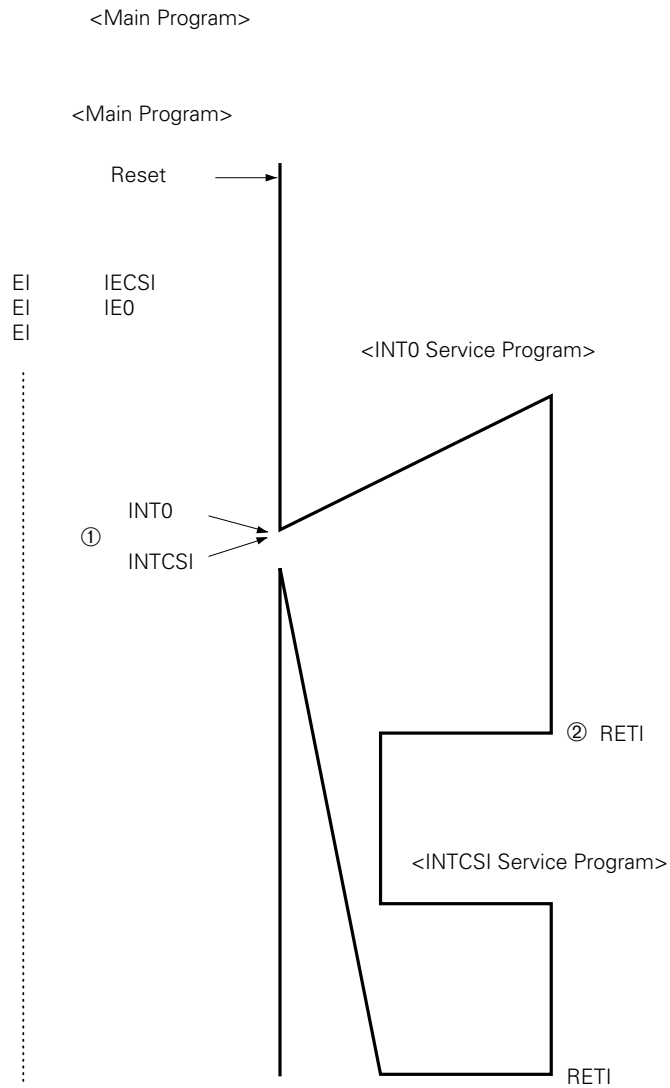
- ① All interrupts disabled and status 0 set by $\overline{\text{RESET}}$ input.
- ② INTO set to falling edge active.
- ③ Interrupts enabled by EI and EI IE_{xxx} instructions.
- ④ On fall of INTO, INTO interrupt service program is started, status is changed to 1 and all interrupts are disabled.
- ⑤ RETI instruction effects return from interrupt, restores status to 0, and enables interrupts.

(3) Pending interrupt execution - interrupt input in interrupt disabled state



- ① Although INT0 is set in the interrupt disabled state, the interrupt flag is held pending.
- ② The INT0 service program is started at point at which interrupts are enabled by the EI instruction.
- ③ Same as ① .
- ④ The INTCSI service routine is started at the point at which the pending INTCSI is enabled.

(4) Pending interrupt execution



- ① If INT0 and INTCSI are generated simultaneously (during execution of the same instruction), INT0, which has the higher interrupt priority, is executed first (INTCSI is held pending).
- ② When the INT0 service program is ended by the RETI instruction, the pending INTCSI service program is started.

CHAPTER 7. STANDBY FUNCTION

The μ PD75402A has a standby function which can reduce the system power consumption. The standby function has the following two modes:

- STOP mode
- HALT mode

(1) STOP mode

In this mode, the main system clock oscillator is stopped and the whole system stops. The CPU current drain is reduced considerably.

Data memory low voltage (up to $V_{DD} = 2\text{ V}$) hold is also possible. Therefore, this mode is effective when retaining the contents of data memory at an ultra-low current drain.

The μ PD75402A STOP mode cannot be reset by interrupt request. It is reset only by $\overline{\text{RESET}}$ input.

(2) HALT mode

This mode stops the CPU operation clock. The system clock oscillator continues to oscillate. In this mode, the current drain is not reduced as much as the STOP mode, but this mode is effective when desiring to resume processing immediately by interrupt request and when desiring to perform intermittent operation, such as timer operation.

The HALT mode is reset by $\overline{\text{RESET}}$ input or interrupt request.

In either mode, the contents of the registers, flags, and data memory immediately before setting to the standby mode are retained. Because the states of the I/O port output latch and the output buffer are retained, the state of the I/O ports is preprocessed so that the current drain of the entire system is minimum.

- Note**
- 1. Low current, low voltage operation is possible by switching the standby mode and CPU clock. However, in either case, the time described in paragraph 5.2.3 is necessary between operation of the PCC register and selection of the new clock and the start of operation by the new clock after switching. Therefore, when combining the clock switching function and standby mode, set the μ PD75402A to the standby mode after the time required for switching has elapsed.**
 - 2. When using the standby mode, ensure that the consumption current at the input/output is a minimum. In particular, do not leave the input port open : be sure to input a low-level or high-level signal.**

★

7.1 STANDBY MODE SETTING AND OPERATION STATES

Table 7-1 Standby Mode Operation States

		STOP Mode	HALT Mode
Setting instruction		STOP instruction	HALT instruction
Operation state	Clock generator	Clock oscillation stopped	CPU clock ϕ only stopped oscillator (oscillation continues)
	Basic interval timer	Operation stopped	Operation (IRQBT set at basic time interval)
	Serial interface	Operation possible only when external SCK input selected as serial clock	Operation possible
	Clock output circuit	Operation stopped	Output other than CPU clock ϕ possible
	External interrupt	INT2 : Operation possible INT0 : Operation impossible	
	CPU	Operation stopped	
Reset signal		$\overline{\text{RESET}}$ input	Interrupt request signal from operable hardware enabled by interrupt enable flag or $\overline{\text{RESET}}$ input.

The STOP mode is set by STOP instruction and the HALT mode is set by HALT instruction. (The STOP instruction and HALT instruction set bits 3 and 2 of the PCC respectively.)

Always write an NOP instruction after the STOP instruction or HALT instruction.

When the CPU operation clock is changed by means of the low-order two bits of the PCC, a time lag may be generated between rewriting of the PCC and changing of the CPU clock. Therefore, when changing the operating clock before the standby mode and when changing the CPU clock after standby mode reset, set the standby mode after the number of machine cycles required to change the CPU clock has elapsed after the PCC is rewritten.

In the standby mode, the data of the general register, flags, mode registers, output latch, and all the other registers which stop operating in the standby mode and the data memory is retained.

Notes are given below.

- Note**
1. When the STOP mode is set, the X1 pin is shorted internally to Vss (GND potential) to suppress clock oscillator leakage. Therefore, do not use the STOP mode with systems that use an external clock.
 2. STOP mode reset by interrupt request differs as follows for the $\mu\text{PD75402A}$ and the evachip installed on the evaluation board:
 - $\mu\text{PD75402A}$ ----- STOP mode not reset by interrupt request.
 - Evachip ----- STOP mode reset by interrupt request.
 To eliminate the affect of this difference, disable all interrupt requests before setting the $\mu\text{PD75402A}$ to the STOP mode.
 3. From the standpoint that an interrupt request signal is used to reset the HALT mode, when there is an interrupt source which sets both the interrupt request flag and interrupt enable flag (1), the HALT mode, even if entered, is immediately reset.

7.2 STANDBY MODE RESET

The STOP mode is reset only by $\overline{\text{RESET}}$ input. The HALT mode is reset by standby release signal by setting of an interrupt request flag enabled by the interrupt enable flag and by $\overline{\text{RESET}}$ input.

The standby mode reset operation is shown in Fig. 7-1.

Note When a standby mode (STOP/HALT) was reset by $\overline{\text{RESET}}$ input, the $\mu\text{PD75402A}$ does not insert a wait before the start of instruction execution. Instruction execution begins simultaneously with resetting of the standby mode.

(1) STOP mode reset by $\overline{\text{RESET}}$ input

When the $\overline{\text{RESET}}$ input drops from high to low, the oscillator begins to oscillate simultaneously with entry into the reset state.

When the $\overline{\text{RESET}}$ input level returns from low to high, instruction execution begins even if oscillation is unstable. Therefore, provide an oscillation stabilization time by making the $\overline{\text{RESET}}$ input low level sufficiently wide.

When the reset state is released, the program branches to the reset start address.

This is different from normal reset operation because the contents of data memory before STOP mode setting are retained.

(2) HALT mode reset by $\overline{\text{RESET}}$ input

When the $\overline{\text{RESET}}$ input drops from high to low, the HALT mode is reset and the $\mu\text{PD75402A}$ enters the reset state.

When the $\overline{\text{RESET}}$ input level returns from low to high, the program branches to the reset start address and instruction execution begins.

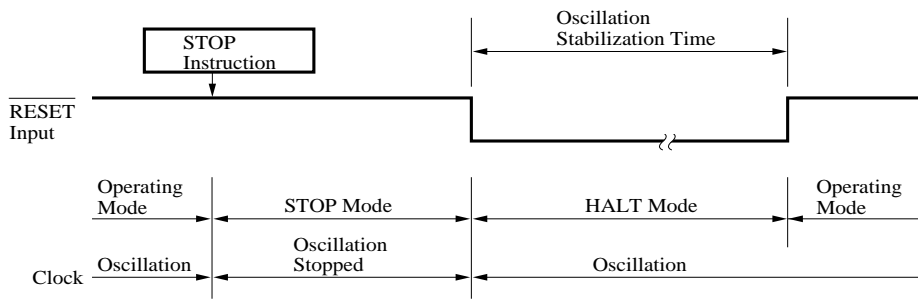
This is different from the normal reset operation because the contents of data memory before HALT mode setting are retained.

(3) HALT mode reset by interrupt generation

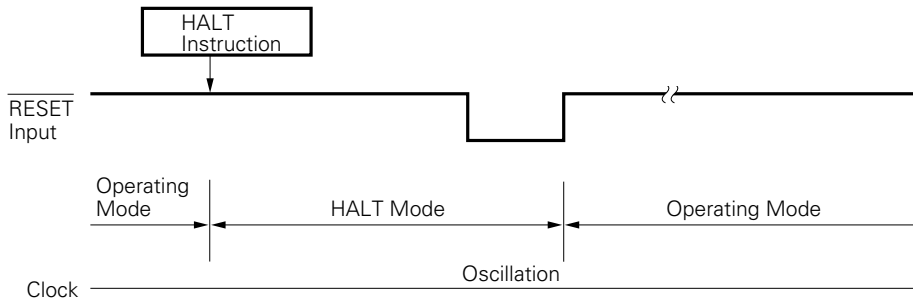
When an interrupt request flag enabled by interrupt enable flag is set (1), a standby release signal is generated and the HALT mode is reset. However, the INT0 interrupt request does not generate the standby release signal.

Fig. 7-1 Standby Mode Reset Operation

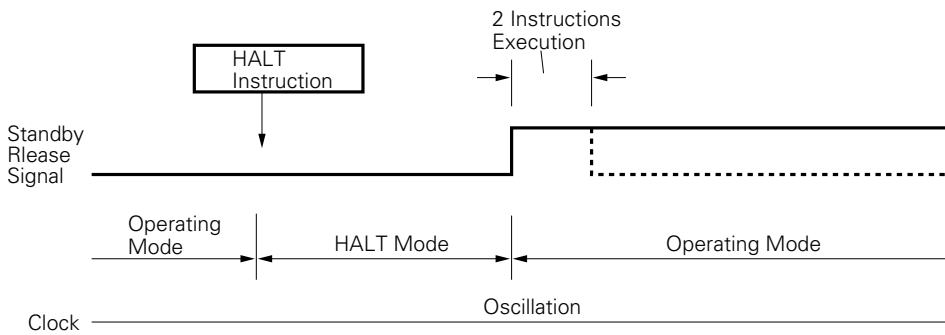
(a) STOP mode reset by $\overline{\text{RESET}}$ input



(b) HALT mode reset by $\overline{\text{RESET}}$ input



(c) HALT mode reset by interrupt generation



Remarks The broken line is for the case when an interrupt request that reset the HALT mode was accepted (IME = 1).

7.3 OPERATION AFTER STANDBY MODE RESET

- (1) When the standby mode was reset by $\overline{\text{RESET}}$ input, normal reset operation is executed. (STOP and HALT modes)
- (2) When the standby mode was reset by interrupt request generation, whether or not a vector interrupt is executed when the CPU resumes instruction execution is determined by the contents of the interrupt master enable flag (IME). (HALT mode)

(a) IME = 0

After HALT mode reset, execution is resumed from the instruction (NOP instruction) after the HALT mode setting instruction.

The interrupt request flag is held.

(b) IME = 1

After HALT mode is reset, a vector interrupt is executed two instructions after the HALT mode setting instruction. However, because a vector interrupt is not generated when the HALT mode was reset by INT2 (testable input), the same processing as (a) is performed.

7.4 STANDBY MODE APPLICATION

When using the standby mode, proceed as follows:

- ① Power interruption or other standby mode setting cause detection by interrupt input or port input.
- ② I/O port processing (Process for minimum current drain)
In particular, do not leave the input port open : be sure to input a low-level or high-level signal. ★
- ③ Specification of interrupt which resets the standby mode (However, for HALT mode, interrupt enable flags which is not reset are cleared.)
- ④ Specification of operation after reset (IME is operated according to whether or not interrupt processing is performed after HALT mode resetting.)
- ⑤ Specification of CPU clock after reset (For switching, provide the number of machine cycles necessary up to standby mode setting.)
- ⑥ Standby mode setting (STOP and HALT instructions)

CHAPTER 8. RESET FUNCTION

When low level is input to the $\overline{\text{RESET}}$ pin, system reset is applied and the hardware enters the state shown in Table 8-1.

When the $\overline{\text{RESET}}$ input goes from low level to high level, the reset state is released. Then, the contents of the lower-order three bits of address 000H of the reset vector table are set into program counter (PC) bits 10 to 8 and the contents of the low-order three bits of address 001H are set into PC bits 7 to 0 and the program branches and begins executing from that branch address. Therefore, reset and starting from an arbitrary address is possible.

Initialize the contents of each register as required in the program.

The $\overline{\text{RESET}}$ pin is a Schmitt-triggered input with hysteresis characteristics at the threshold level. To prevent misoperation by noise, a function which rejects narrow band noise by analog delay is also provided on the chip (see Fig. 8-1).

For reset operation at power-on, provide an ample oscillation stabilization time from power-on to reset signal acceptance as shown in Fig. 8-2.

Fig. 8-1 Reset Signal Acceptance

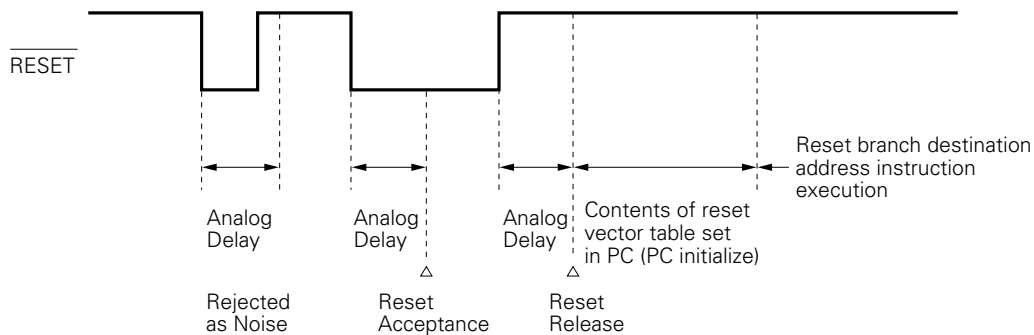


Fig. 8-2 Reset at Power-on

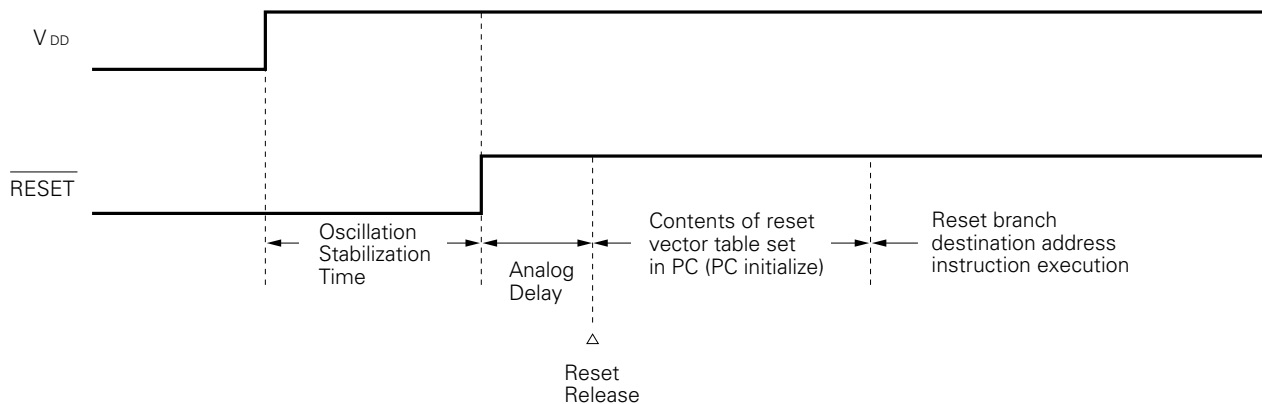


Table 8-1 State of Hardware after Reset

Hardware		$\overline{\text{RESET}}$ Input standby mode	$\overline{\text{RESET}}$ Input during operation
Program counter (PC)		Low-order 3 bits of program memory address 000H set in PC10 to PC8 and contents of address 001H set in PC7 to PC0	Low-order 3 bits of program memory address 000H set in PC10 to PC8 and contents of address 001H set in PC7 to PC0
PSW	Carry flag (CY)	Retained	Undefined
	Skip flag (SK0 to SK2)	0	0
	Interrupt status flag (IST0)	0	0
Stack pointer (SP)		Undefined	Undefined
Data Memory (RAM)		Retained*	Undefined
General register (X, A, H, L)		Retained	Undefined
Basic interval timer	Counter (BT)	Undefined	Undefined
	Mode register (BTM)	0	0
Serial interface	Shift register (SIO)	Retained	Undefined
	Operation mode register (CSIM)	0	0
	SBI control register (SBIC)	0	0
	Slave address register (SVA)	Retained	Undefined
Clock generator, clock output circuit	Processor clock control register (PCC)	0	0
	Clock output mode register (CLOM)	0	0
Interrupt function	Interrupt request flag (IRQ _{xxx})	Reset (0)	Reset (0)
	Interrupt enable flag (IE _{xxx})	0	0
	Interrupt master enable flag (IME)	0	0
	INT0 mode register (IM0)	0	0
Digital input/output port	Output buffer	OFF	OFF
	Output latch	Clear (0)	Clear (0)
	I/O mode register (PMGA, PMGB)	0	0
	Pull-up resistor specification register (POGA)	0	0
Pin state	Pin state P00 to P03, P10, P12, P20 to P23, P30 to P33, P60 to P63	Input	Input
	P50 to P53	<ul style="list-style-type: none"> • On-chip pull-up resistor •••• High level • Open-drain •••• High impedance 	<ul style="list-style-type: none"> • On-chip pull-up resistor •••• High level • Open-drain •••• High impedance

* The contents of data memory addresses 038H to 03DH are made undefined by $\overline{\text{RESET}}$ input.

CHAPTER 9. INSTRUCTION SET

The 75X series instruction set is an improved and expanded version of old μ PD7500 series instruction set. It is a revolutionary new instruction set which retains succession from the μ PD7500 series. The μ PD75402A instruction set is a 75X instruction subset, and has the following features:

- (1) Multipurpose bit manipulation instruction
- (2) Efficient 4-bit manipulation instruction
- (3) 8-bit data transfer instruction
- (4) Stack instructions and base correction instructions with increase program efficiency
- (5) Table reference instructions suitable for continuous reference
- (6) 1-byte relative branch instruction
- (7) Easy to understand NEC standard mnemonics

For the addressing modes which can be used when operating the data memory, see **CHAPTER 3 "FEATURES OF ARCHITECTURE AND MEMORY MAP"**.

9.1 SPECIAL INSTRUCTIONS

This section outlines the special instructions of the μ PD75402A instruction set.

9.1.1 Bit Manipulation Instructions

μ PD75402A bit manipulation can be performed by various instructions, such as the following:

- | | | |
|------------------------|--------|---------------|
| (a) Bit set : | SET1 | mem. bit |
| | SET1 | fmem. bit |
| (b) Bit clear: | CLR1 | mem. bit |
| | CLR1 | fmem. bit |
| (c) Bit test : | SKT | mem. bit |
| | SKT | fmem. bit |
| (d) Bit test : | SKF | mem. bit |
| | SKF | fmem. bit |
| (e) Bit test & clear : | SKTCLR | fmem. bit |
| (f) Boolean operation: | AND1 | CY, fmem. bit |
| | OR1 | CY, fmem. bit |
| | XOR1 | CY, fmem. bit |

fmem. bit is the bit address specified by special address bit manipulation addressing.

Especially, since I/O ports can always use all the bit manipulation instructions above, I/O port operation can be performed very efficiently.

9.1.2 Stack Instructions

The following two kinds of stack instructions are available with the μ PD75402A.

- (a) MOV A, #n4 or MOV XA, #n8
- (b) MOV HL, #n8

“Stack” signifies that these two kinds of instructions are placed in contiguous addresses.

Example A0 : MOV A, #0
 A1 : MOV A, #1
 XA7: MOV XA, #07

When stack instructions are stacked such as in the example above, when the address executed first is A0, it is executed by replacing the next two instructions with NOP instructions. When the address executed first is A1, it is executed by replacing the next instruction with an NOP instruction. That is, only the instruction executed first is effective, all the stack instructions following it are processed as NOP instructions.

Constants can be efficiently set to accumulator (A register, register pair XA) and data pointer (register pair HL) by using these stack instructions.

9.1.3 Base Correction Instructions

Depending on the application, the result of addition of 4-bit data must be converted to decimal numbers or to base-6, such as time.

Base correction instructions for converting the result of addition of 4-bit data to an arbitrary base are available with the μ PD75402A instruction set.

(a) Base correction at addition

If the base value to be corrected is made m , accumulator and memory (HL) are added and the sum is converted to base- m by combination:

```
ADDS A, #16-m
```

```
ADDC A, @HL ; A, CY ← A + (HL) + CY
```

```
ADDS A, #m
```

Overflow remains in the carry flag.

When a carry is output as a result of execution of the `ADDC A, @HL` instruction, the following `ADDS A, #n4` instruction is skipped. If carry is not output, the `ADDS A, #n4` instruction is executed. At this time, this instruction skip function is disabled and the next instruction is not skipped even if carry is output as the result of addition. Therefore, the program can continue after the `ADDS A, #n4` instruction.

Example Decimal add accumulator and memory.

```
ADDS A, #6
```

```
ADDC A, @HL ; A, CY ← A + (HL) + CY
```

```
ADDS A, #10
```

```
⋮
```

9.1.4 Skip Instruction and Number of Machine Cycles Required by Skip

With the μ PD75402A instruction set, a program is formed by condition judgment by skip.

If the skip condition is satisfied when a skip instruction is executed, the following instruction is skipped and the instruction after the instruction is executed.

When a skip was generated, one machine cycle is required to skip.

9.2 INSTRUCTION SET AND ITS OPERATION

(1) Operation identifier and description

The operands are described in the operand field of each instruction in accordance with the description for the operand identifier of the instruction. (See "RA 75X Assembler Package User's Manual Language Volume (EEU-730) for details.) For parameters with multiple elements in the description, one of the elements is selected. Upper case letters and the symbols #, @, !, and \$ are key words and described unchanged.

For immediate data, a suitable value or label is described.

Instead of mem, fmem, bit, etc., various kinds of registers and flag symbols shown in Table 3-4 can be written as labels (however, in the case of fmem there are restrictions on the labels that can be written. See Table 3-3 "Applicable Addressing Modes at Peripheral Hardware Operation" and Table 3-4 "μPD75402A I/O Map" for details.

Identifier	Description
reg	X, A, H, L
reg1	X, H, L
rp	XA, HL
n4	4-bit immediate data or label
n8	8-bit immediate data or label
mem	8-bit immediate data or label*
bit	2-bit immediate data or label
fmem	FB0H to FBFH, FF0H to FFFH immediate data or label
addr	11-bit immediate data or label
caddr	11-bit immediate data or label
faddr	11-bit immediate data or label
PORTn	PORT0 to PORT3, PORT5, PORT6
IExxx	IEBT, IECSI, IE0, IE2

* For 8-bit data processing, mem can describe even address only.

(2) Operation description legend

- A : A register; 4-bit accumulator
- H : H register
- L : L register
- X : X register
- XA : Register pair (XA); 8-bit accumulator
- HL : Register pair (HL)
- PC : Program counter
- SP : Stack pointer
- CY : Carry flag; bit accumulator
- PSW : Program status word
- PORTn : Port n (n = 0 to 3, 5, 6)
- IME : Interrupt master enable flag
- IExxx : Interrupt enable flag
- PCC : Processor clock control register
- . : Address, bit delimiter
- (xx) : Contents addressed by xx
- xxH : Hexadecimal data

(3) Description of addressing area field symbols

* 1	MB = 0	Data memory addressing
* 2	MB = 0 (00H to 3FH) MB = 15 (80H to FFH)	
* 3	MB = 15, fmem = FB0H to FBFH, FF0H to FFFH	
* 4	addr = 000H to 77FH	Program memory addressing
* 5	addr = (Current PC) - 15 to (Current PC) - 1, (Current PC) + 16 to (Current PC) + 2	
* 6	caddr = 000H to 77FH	
* 7	faddr = 000H to 77FH	

- Remarks**
1. MB is the accessible memory bank.
 2. *4 to *7 are the addressable areas.

(4) Description of machine cycle field

S is the number of machine cycles required when the skip operation is performed by an instruction with skip.

The value of S changes as follows:

- Do not skip next instruction S = 0
- Skip next instruction S = 1

One machine cycle equals one cycle of CPU clock ϕ . Three times can be selected by PCC setting. (See section 5.2.2(1).)

Note 1	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Move instructions	MOV	A, #n 4	1	1	$A \leftarrow n\ 4$		Stack A
		XA, #n 8	2	2	$XA \leftarrow n\ 8$		Stack A
		HL, #n 8	2	2	$HL \leftarrow n\ 8$		Stack B
		A, @HL	1	1	$A \leftarrow (HL)$	*1	
		@HL, A	1	1	$(HL) \leftarrow A$	*1	
		A, mem	2	2	$A \leftarrow (mem)$	*2	
		XA, mem	2	2	$XA \leftarrow (mem)$	*2	
		mem, A	2	2	$(mem) \leftarrow A$	*2	
	XCH	A, @HL	1	1	$A \leftrightarrow (HL)$	*1	
		A, mem	2	2	$A \leftrightarrow (mem)$	*2	
		XA, mem	2	2	$XA \leftrightarrow (mem)$	*2	
		A, reg1	1	1	$A \leftrightarrow reg1$		
	MOVT	XA, @PCXA	1	3	$XA \leftarrow (PC_{10-8} + XA)_{ROM}$		
Arithmetic and logic instructions	ADDS	A, #n 4	1	1 + S	$A \leftarrow A + n\ 4$		carry
		A, @HL	1	1 + S	$A \leftarrow A + (HL)$	*1	carry
	ADDC	A, @HL	1	1	$A, CY \leftarrow A + (HL) + CY$	*1	
	AND	A, @HL	1	1	$A \leftarrow A \wedge (HL)$	*1	
	XOR	A, @HL	1	1	$A \leftarrow A \vee (HL)$	*1	
Note 2	RORC	A	1	1	$CY \leftarrow A_0, A_3 \leftarrow CY, A_{n-1} \leftarrow A_n$		
	NOT	A	2	2	$A \leftarrow \bar{A}$		
Note 3	INCS	reg	1	1 + S	$reg \leftarrow reg + 1$		reg = 0
		mem	2	2 + S	$(mem) \leftarrow (mem) + 1$	*2	(mem) = 0
	DECS	reg	1	1 + S	$reg \leftarrow reg - 1$		reg = FH
Note 4	SKE	reg, #n 4	2	2 + S	Skip if reg = n 4		reg = n4
		A, @HL	1	1 + S	Skip if A = (HL)	*1	A = (HL)
Carry flag operation instructions	SET 1	CY	1	1	$CY \leftarrow 1$		
	CLR 1	CY	1	1	$CY \leftarrow 0$		
	SKT	CY	1	1 + S	Skip if CY = 1		CY = 1
	NOT 1	CY	1	1	$CY \leftarrow \bar{CY}$		

- Note**
1. Instruction Group
 2. Accumulator operation instructions
 3. Increment/decrement instructions
 4. Compare instructions

CHAPTER 9. INSTRUCTION SET

Note	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Bit manipulation instructions	SET 1	mem. bit	2	2	$(\text{mem. bit}) \leftarrow 1$	*2	
		f mem. bit	2	2	$(\text{f mem. bit}) \leftarrow 1$	*3	
	CLR 1	mem. bit	2	2	$(\text{mem. bit}) \leftarrow 0$	*2	
		f mem. bit	2	2	$(\text{f mem. bit}) \leftarrow 0$	*3	
	SKT	mem. bit	2	2 + S	Skip if $(\text{mem. bit}) = 1$	*2	$(\text{mem. bit}) = 1$
		f mem. bit	2	2 + S	Skip if $(\text{f mem. bit}) = 1$	*3	$(\text{f mem. bit}) = 1$
	SKF	mem. bit	2	2 + S	Skip if $(\text{mem. bit}) = 0$	*2	$(\text{mem. bit}) = 0$
		f mem. bit	2	2 + S	Skip if $(\text{f mem. bit}) = 0$	*3	$(\text{f mem. bit}) = 0$
	SKTCLR	f mem. bit	2	2 + S	Skip if $(\text{f mem. bit}) = 1$ and clear	*3	$(\text{f mem. bit}) = 1$
	AND 1	CY, f mem. bit	2	2	$\text{CY} \leftarrow \text{CY} \wedge (\text{f mem. bit})$	*3	
OR 1	CY, f mem. bit	2	2	$\text{CY} \leftarrow \text{CY} \vee (\text{f mem. bit})$	*3		
XOR 1	CY, f mem. bit	2	2	$\text{CY} \leftarrow \text{CY} \nabla (\text{f mem. bit})$	*3		
Branch instructions	BR	addr	-	-	$\text{PC}_{10-0} \leftarrow \text{addr}$ <div style="border: 1px solid black; padding: 5px; display: inline-block;"> The assembler selects the optimum instruction from among BRCB!, caddr, and BR\$ addr. </div>	*4	
		\$addr	1	2	$\text{PC}_{10-0} \leftarrow \text{addr}$	*5	
	BRCB	! caddr	2	2	$\text{PC}_{10-0} \leftarrow \text{caddr}$	*6	
Subroutine stack control instructions	CALLF	! faddr	2	2	$(\text{SP}-4) (\text{SP}-1) (\text{SP}-2) \leftarrow 0, \text{PC}_{10-0}$ $(\text{SP}-3) \leftarrow 0000$ $\text{PC}_{10-0} \leftarrow \text{faddr}, \text{SP} \leftarrow \text{SP}-4$	*7	
	RET		1	3	$\text{PC}_{10-0} \leftarrow (\text{SP}) (\text{SP}+3) (\text{SP}+2)$ $\text{SP} \leftarrow \text{SP}+4$		
	RETS		1	3 + S	$\text{PC}_{10-0} (\text{SP}) (\text{SP}+3) (\text{SP}+2)$ $\text{SP} \leftarrow \text{SP}+4,$ then skip unconditionally		None
	RETI		1	3	$\text{PC}_{10-0} \leftarrow (\text{SP}) (\text{SP}+3) (\text{SP}+2)$ $\text{PSW} \leftarrow (\text{SP}+4) (\text{SP}+5),$ $\text{SP} \leftarrow \text{SP}+6$		
	PUSH	rp	1	1	$(\text{SP}-1) (\text{SP}-2) \leftarrow \text{rp}, \text{SP} \leftarrow \text{SP}-2$		
	POP	rp	1	1	$\text{rp} \leftarrow (\text{SP}+1) (\text{SP}), \text{SP} \leftarrow \text{SP}+2$		
Interrupt control instructions	EI		2	2	$\text{IME} (\text{IPS. } 3) \leftarrow 1$		
		IE _{xxx}	2	2	$\text{IE}_{\text{xxx}} \leftarrow 1$		
	DI		2	2	$\text{IME} (\text{IPS. } 3) \leftarrow 0$		
		IE _{xxx}	2	2	$\text{IE}_{\text{xxx}} \leftarrow 0$		

Note Instruction Group

Note 1	Mnemonic	Operand	Bytes	Machine Cycle	Operation	Addressing Area	Skip Condition
Note 2	IN	A, PORT _n	2	2	$A \leftarrow \text{PORT}_n$ ($n = 0 - 3, 5, 6$)		
	OUT	PORT _n , A	2	2	$\text{PORT}_n \leftarrow A$ ($n = 2, 3, 5, 6$)		
CPU control instructions	HALT		2	2	Set HALT Mode (PCC.2 \leftarrow 1)		
	STOP		2	2	Set STOP Mode (PCC.3 \leftarrow 1)		
	NOP		1	1	No Operation		

- Note**
1. Instruction Group
 2. I/O instructions

9.3 OPERATION CODE OF EACH INSTRUCTION

(1) Description of operation code symbols

R ₁	R ₀	reg
0	0	A
0	1	X
1	0	L
1	1	H

↑
reg
↓

↑
reg 1
↓

P ₁	reg-pair
0	XA
1	HL

↑
rp
↓

N ₂	N ₁	N ₀	IE _{xxx}
0	0	0	IEBT
1	0	1	IECSI
1	1	0	IE0
1	1	1	IE2

- In : Immediate data for n4, n8
- Dn : Immediate data for mem
- Bn : Immediate data for bit
- Nn : Immediate data for n, IE_{xxx}
- An : Immediate data for [relative address distance with branch address (2 to 16)] - 1
- Sn : Immediate data for one's complement of [relative address distance with branch address (15 to 1)]

(2) Bit manipulation addressing operation code

bit-addr of the second byte of the operation code of an instruction with fmem. bit at the operands is shown below.

bit-addr								Accessible bits
1	0	B ₁	B ₀	F ₃	F ₂	F ₁	F ₀	Operable bits of FB0H to FBFH
1	1	B ₁	B ₀	F ₃	F ₂	F ₁	F ₀	Operable bits of FF0H to FFFH

- Bn : Immediate data for bit address (0 to 3) described at bit
- Fn : Immediate data for low-order four bits of address described at fmem

CHAPTER 9. INSTRUCTION SET

Note 1	Mnemonic	Operand	Operation Code															
			B ₁								B ₂							
Move instructions	MOV	A, #n 4	0	1	1	1	I ₃	I ₂	I ₁	I ₀								
		rp, #n 8	1	0	0	0	1	0	P ₁	1	I ₇	I ₆	I ₅	I ₄	I ₃	I ₂	I ₁	I ₀
		A, @HL	1	1	1	0	0	0	0	1								
		@HL, A	1	1	1	0	1	0	0	0								
		A, mem	1	0	1	0	0	0	1	1	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
		XA, mem	1	0	1	0	0	0	1	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	0
		mem, A	1	0	0	1	0	0	1	1	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
	mem, XA	1	0	0	1	0	0	1	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	0	
	XCH	A, @HL	1	1	1	0	1	0	0	1								
		A, mem	1	0	1	1	0	0	1	1	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
XA, mem		1	0	1	1	0	0	1	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	0	
A, reg1		1	1	0	1	1	0	R ₁	R ₀									
MOVT	XA, @PCXA	1	1	0	1	0	0	0	0									
Arithmetic and logic instructions	ADDS	A, #n 4	0	1	1	0	I ₃	I ₂	I ₁	I ₀								
		A, @HL	1	1	0	1	0	0	1	0								
	ADDC	A, @HL	1	0	1	0	1	0	0	1								
	AND	A, @HL	1	0	0	1	0	0	0	0								
	OR	A, @HL	1	0	1	0	0	0	0	0								
	XOR	A, @HL	1	0	1	1	0	0	0	0								
Note 2	RORC	A	1	0	0	1	1	0	0	0								
	NOT	A	1	0	0	1	1	0	0	1	0	1	0	1	1	1	1	1
Note 3	INCS	reg	1	1	0	0	0	0	R ₁	R ₀								
		mem	1	0	0	0	0	0	1	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
	DECS	reg	1	1	0	0	1	0	R ₁	R ₀								
Note 4	SKE	reg, #n 4	1	0	0	1	1	0	1	0	I ₇	I ₆	I ₅	I ₄	0	0	R ₁	R ₀
		A, @HL	1	0	0	0	0	0	0	0								
Carry flag operation instructions	SET 1	CY	1	1	1	0	0	1	1	1								
	CLR 1	CY	1	1	1	0	0	1	1	0								
	SKT	CY	1	1	0	1	0	1	1	1								
	NOT 1	CY	1	1	0	1	0	1	1	0								

- Note**
1. Instruction Group
 2. Accumulator operation instructions
 3. Increment/decrement instructions
 4. Compare instruction

Note 1	Mnemonic	Operand	Operation Code																
			B ₁								B ₂								
Memory bit manipulation instructions	SET 1	mem. bit	1	0	B ₁	B ₀	0	1	0	1	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
		f mem. bit	1	0	0	1	1	1	0	1	bit-addr								
	CLR 1	mem. bit	1	0	B ₁	B ₀	0	1	0	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
		f mem. bit	1	0	0	1	1	1	0	0	bit-addr								
	SKT	mem. bit	1	0	B ₁	B ₀	0	1	1	1	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
		f mem. bit	1	0	1	1	1	1	1	1	bit-addr								
	SKF	mem. bit	1	0	B ₁	B ₀	0	1	1	0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	0	
		f mem. bit	1	0	1	1	1	1	1	0	bit-addr								
	SKTCLR	f mem. bit	1	0	0	1	1	1	1	1	bit-addr								
AND 1	CY, f mem. bit	1	0	1	0	1	1	0	0	bit-addr									
OR 1	CY, f mem. bit	1	0	1	0	1	1	1	0	bit-addr									
XOR 1	CY, f mem. bit	1	0	1	1	1	1	0	0	bit-addr									
Note 2	BR	\$addr <small>(+16) } (+2) (-1) (-15)</small>	0	0	0	0	A ₃	A ₂	A ₁	A ₀									
			1	1	1	1	S ₃	S ₂	S ₁	S ₀									
	BRCB	! caddr	0	1	0	1	0	←				caddr →							
Subroutine and stack instructions	CALLF	! faddr	0	1	0	0	0	←				faddr →							
	RET		1	1	1	0	1	1	1	0									
	RETS		1	1	1	0	0	0	0	0									
	RETI		1	1	1	0	1	1	1	1									
	PUSH	rp	0	1	0	0	1	0	P ₁	1									
	POP	rp	0	1	0	0	1	0	P ₁	0									
Interrupt control instructions	EI		1	0	0	1	1	1	0	1	1	0	1	1	0	0	1	0	
		IE _{xxx}	1	0	0	1	1	1	0	1	1	0	0	1	1	N ₂	N ₁	N ₀	
	DI		1	0	0	1	1	1	0	0	1	0	1	1	0	0	1	0	
		IE _{xxx}	1	0	0	1	1	1	0	0	1	0	0	1	1	N ₂	N ₁	N ₀	
Note 3	IN	A, PORT _n	1	0	1	0	0	0	1	1	1	1	1	1	0	N ₂	N ₁	N ₀	
	OUT	PORT _n , A	1	0	0	1	0	0	1	1	1	1	1	1	0	N ₂	N ₁	N ₀	
Note 4	HALT		1	0	0	1	1	1	0	1	1	0	1	0	0	0	1	1	
	STOP		1	0	0	1	1	1	0	1	1	0	1	1	0	0	1	1	
	NOP		0	1	1	0	0	0	0	0									

- Note**
1. Instruction Group
 2. Branch instructions
 3. I/O instructions
 4. CPU control instructions

9.4 INSTRUCTION FUNCTIONS AND APPLICATION

9.4.1 Move Instructions

MOV A, #n4

Function: $A \leftarrow n4$; $n4 = l_3 \text{ to } l_0 : 0 \text{ to } FH$

Moves 4-bit immediate data $n4$ to the A register (4-bit accumulator).

This instruction has a stacking effect (group A). When placed after a MOV A, #n4 or MOV XA, #n8 instruction, stack instructions following the executed instruction are processed as NOP.

Application examples: ① Set 0BH into accumulator.

```
MOV A, #0BH
```

② Select the data to be output at Port 3 from 0 to 2.

```
A0: MOV A, #0
```

```
A1: MOV A, #1
```

```
A2: MOV A, #2
```

```
OUT PORT 3, A
```

MOV rp, #n8

Function: $rp \leftarrow n8$; $n8 = l_7 \text{ to } l_0 : 00H \text{ to } FFH$

Moves 8-bit immediate data $n8$ to register pair rp (XA, HL).

This instruction has a stacking effect. There are two stacking effects: Group A (MOV A, #n4 instruction and MOV XA, #n8 instruction), and group B (MOV HL, #n8 instruction). When instructions of the same group are placed consecutively, the stack instructions after the executed instruction are processed as NOP.

Application example: Set 5FH into register pair HL.

```
MOV HL, #5FH
```

MOV A, @HL

Function: $A \leftarrow (HL)$

Moves the data memory contents addressed by the contents of register pair HL to the A register.

Application example: Move the data of address 3EH to the A register.

```
MOV HL, #3EH
```

```
MOV A, @HL
```

MOV @HL, A

Function: $(HL) \leftarrow A$

Moves the contents of the A register to the data memory addressed by the contents of register pair HL.

MOV A, mem

Function: $A \leftarrow (\text{mem}); \text{mem} = D_7 \text{ to } D_0 : 00\text{H to } 3\text{FH}$

Moves the data memory contents addressed by 8-bit immediate data mem to the A register.

MOV XA, mem

Function: $A \leftarrow (\text{mem}), X \leftarrow (\text{mem} + 1); \text{mem} = D_7 \text{ to } D_0 : 00\text{H to } 3\text{EH}$

Moves the data memory contents addressed by 8-bit immediate data mem to the A register and the contents of the next address to the X register.

mem can specify even addresses.

Application example: Move the data of addresses 20H and 21H to register pair XA.

```
MOV XA, 20H
```

MOV mem, A

Function: $(\text{mem}) \leftarrow A; \text{mem} = D_7 \text{ to } D_0 : 00\text{H to } 3\text{FH}$

Move the contents of the A register to the data memory addressed by 8-bit immediate data mem.

MOV mem, XA

Function: $(\text{mem}) \leftarrow A, (\text{mem} + 1) \leftarrow X; \text{mem} = D_7 \text{ to } D_0 : 00\text{H to } 3\text{EH}$

Moves the contents of the A register to the data memory addressed by 8-bit immediate data mem and the contents of the X register to the next memory address.

mem can specify even addresses.

XCH A, @HL

Function: $A \leftarrow (HL)$

Exchanges the contents of the A register and the contents of the data memory addressed by the contents of register pair HL.

Application example: Exchange the data of data memory addresses 20H to 2FH and the data of addresses 30H to 3FH.

```

        MOV HL, #30H
LOOP:  XCH  A, @HL    ; A ↔ (3×)
        MOV  H, #2
        XCH  A, @HL    ; A ↔ (2×)
        MOV  H, #3
        XCH  A, @HL    ; A ↔ (3×)
        INCS L        ; L ← L + 1
        BR   LOOP

```

XCH A, mem

Function: $A \leftrightarrow (\text{mem})$; mem = D₇ to D₀ ; 00H to 3FH

Exchanges the contents of the A register and the data memory contents addressed by 8-bit immediate data mem.

XCH XA, mem

Function: $A \leftrightarrow (\text{mem})$, $X \leftrightarrow (\text{mem} + 1)$; mem = D₇ to D₀ : 00H to 3EH

Exchanges the contents of the A register and the data memory contents addressed by 8-bit immediate data mem and exchanges the contents of the X register and the contents of the next memory address. mem can specify even addresses.

XCH A, reg1

Function: $A \leftrightarrow \text{reg1}$

Exchanges the contents of the A register and the contents of register reg1 (X, H, L).

9.4.2 Table Reference Instructions

MOVT XA, @PCXA

Function: $XA \leftarrow \text{ROM}(\text{PC}_{10} \text{ to } \text{PC}_8 + XA)$

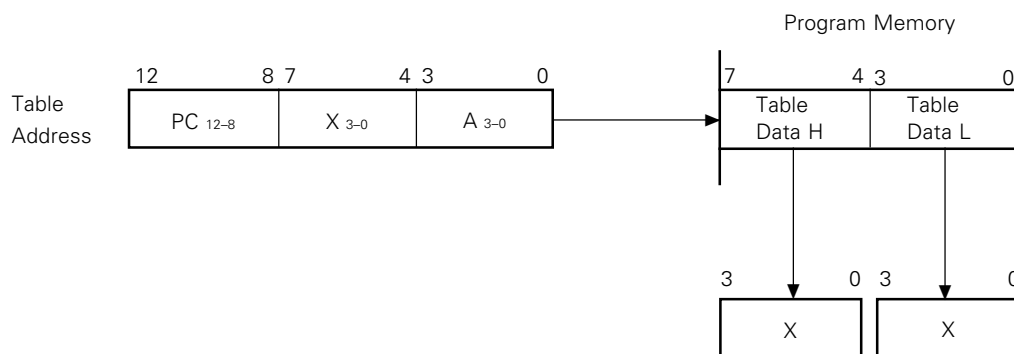
Moves the high-order three bits (PC_{10} to PC_8) of the program counter (PC) and the low-order four bits of the table data in the program memory addressed by the contents of register pair XA to the A register and the high-order four bits to the X register.

The high-order three bits of the table address are determined by the contents of the program counter (PC) when this instruction is executed.

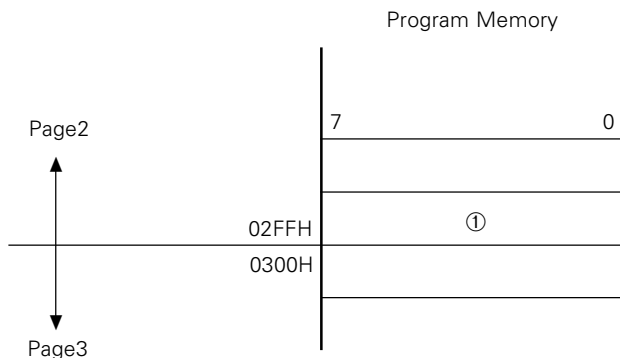
The necessary data must be preprogrammed at the table area by assembler pseudo instruction (DB instruction).

The program counter is not affected by execution of this instruction.

This instruction is effective when referencing table data consecutively.



Note The MOVT XA, @PCXA instruction usually references the table data of the page containing the instruction. However, when the instruction is at address $\times\text{FFH}$, the table data of the next page is referenced instead of the table data on that page.



For instance, when there is a MOVT XA, @PCXA instruction at position ① in the figure above, the table data specified by the contents of register pair XA of page 3 instead of page 2 is moved to register pair XA.

9.4.3 Arithmetic and Logic Instructions

ADDS A, #n4

Function: $A \leftarrow A + n4$; Skip if carry; $n4 = I_3 \text{ to } I_0 : 0 \text{ to } FH$

Binary adds 4-bit immediate data $n4$ to the contents of the A register and skips the next instruction if a carry is generated. The carry flag is not affected.

When combined with an ADDC A, @HL instruction, this instruction becomes a base correction instruction. (See [section 9.1](#).)

ADDS A, @HL

Function: $A \leftarrow A + (HL)$; Skip if carry

Binary adds the data memory contents addressed by register pair HL to the contents of the A register and skips the next instruction if a carry is generated. The carry flag is not affected.

ADDC A, @HL

Function: $A, CY \leftarrow A + (HL) + CY$

Binary adds the data memory contents addressed by register pair HL to the contents of the A register, including the carry flag. If a carry is generated, the carry flag is set. If a carry is not generated, the carry flag is reset.

When an ADDS A, #n4 instruction is placed after this instruction, when a carry is generated at this instruction, the ADDS A, #n4 instruction is skipped. When a carry is not generated, the ADDS A, #n4 instruction is executed and the skip function of the ADDS A, #n4 instruction is disabled. Therefore, the combination of these instructions can be used in base correction (see [section 9.1](#)).

AND A, @HL

Function: $A \leftarrow A \wedge (HL)$

ANDs the contents of the A register and the data memory contents addressed by register pair HL and sets the result into the A register.

OR A, @HL

Function: $A \leftarrow A \vee (HL)$

ORs the contents of the A register and the data memory contents addressed by register pair HL and sets the result into the A register.

XOR A, @HL

Function: $A \leftarrow A \nabla (HL)$

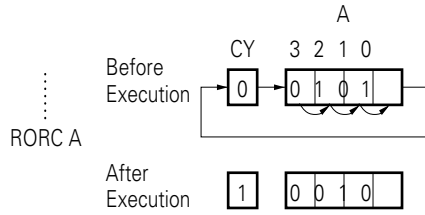
Exclusive-ORs the contents of the A register and the data memory contents addressed by register pair HL and sets the result into the A register.

9.4.4 Accumulator Operation Instructions

RORC A

Function: $CY \leftarrow A_0$ A_n to $A_1 \leftarrow A_n$, $A_3 \leftarrow CY$ ($n = 1$ to 3)

Rotates the contents of the A register (4-bit accumulator), including the carry flag, to the right one bit at a time.



NOT A

Function: $A \leftarrow \bar{A}$

Takes the one's complement (inverts each bit) of the A register (4-bit accumulator).

9.4.5 Increment/Decrement Instructions

INCS reg

Function: $\text{reg} \leftarrow \text{reg} + 1$; Skip if $\text{reg} = 0$

Increments the contents of register reg (X, A, H, L). When the contents of register reg become 0 as the result of incrementing, skips the next instruction.

INCS mem

Functions: $(\text{mem}) \leftarrow (\text{mem}) + 1$; Skip if $(\text{mem}) = 0$, $\text{mem} = D_7$ to D_0 : 00H to FFH

Increments the data memory contents addressed by 8-bit immediate data mem . When the data memory contents become 0 as a result of incrementing, skips the next instruction.

DECS reg

Function: $\text{reg} \leftarrow \text{reg} - 1$; Skip if $\text{reg} = \text{FH}$

Decrements the contents of register reg (X, A, H, L). When the contents of register reg become FH as a result of decrementing, skips the next instruction.

9.4.6 Compare Instructions

SKE reg, #n4

Function: Skip if reg = n4; n4 = l₃ to l₀ : 0 to FH

If the contents of register reg (X, A, H, L) equal 4-bit immediate data n4, skips the next instruction.

SKE A, @HL

Function: Skip if A = (HL)

If the contents of the A register and the data memory contents addressed by register pair HL, skips the next instruction.

9.4.7 Carry Flag Operation Instructions

SET1 CY

Function: $CY \leftarrow 1$

Sets the carry flag.

CLR1 CY

Function: $CY \leftarrow 0$

Clears the carry flag.

SKT CY

Function: Skip if $CY = 1$

When the carry flag is 1, skips the next instruction.

NOT1 CY

Function: $CY \leftarrow \overline{CY}$

Inverts the carry flag. If the carry flag is 0, it becomes 1 and if it is 1, it becomes 0.

9.4.8 Bit Manipulation Instructions

SET1 mem. bit

Function: (mem. bit) \leftarrow 1; mem = D₇ to D₀ : 00H to 3FH, bit = B₁ to B₀ : 0 to 3

Sets the bit specified by 2-bit immediate data bit of the address specified by 8-bit immediate data mem.

SET1 fmem. bit

Function: (bit specified by operand) \leftarrow 1

Sets the data memory bit specified by bit manipulation addressing (fmem. bit).

CLR1 mem. bit

Function: (mem. bit) \leftarrow 0; mem = D₇ to D₀ : 00H to 3FH, bit = B₁ to B₀ : 0 to 3

Clears the bit specified by 2-bit immediate data bit of the address specified by 8-bit immediate data mem.

CLR1 fmem. bit

Function: (bit specified by operand) \leftarrow 0

Clears the data memory bit specified by bit manipulation addressing (fmem. bit).

SKT mem. bit

Function: Skip if (mem. bit) = 1; mem = D₇ to D₀ : 00H to 3FH, bit = B₁ to B₀ : 0 to 3

If 2-bit immediate data bit of the address specified by 8-bit immediate data mem is 1, skips the next instruction.

SKT fmem. bit

Function: Skip if (bit specified by operand) = 1

If the data memory bit specified by bit manipulation addressing (fmem. bit) is 1, skips the next instruction.

SKF mem. bit

Function: Skit if (mem. bit) = 0; mem = D₇ to D₀ : 00H to 3FH, bit = B₁ to B₀ : 0 to 3

If the bit specified by 2-bit immediate data bit of the address specified by 8-bit immediate data mem is 0, skips the next instruction.

SKF fmem. bit

Function: Skip if (bit specified by operand) = 0

If the contents of the data memory bit specified by bit manipulation addressing (fmem. bit) is 0, skips the next instruction.

SKTCLR fmem. bit

Function: Skip if (bit specified by operand) = 1 then clear

If the data memory bit specified by bit manipulation addressing (fmem. bit) is 1, skips the next instruction and clears that bit to 0.

AND1 CY, fmem. bit

Function: $CY \leftarrow CY \wedge (\text{bit specified by operand})$

ANDs the contents of the carry flag and the contents of the data memory bit specified by bit manipulation addressing (fmem. bit) and sets the result into the carry flag.

OR1 CY, fmem. bit

Function: $CY \leftarrow CY \vee (\text{bit specified by operand})$

ORs the contents of the carry flag and the contents of the data memory bit specified by bit manipulation addressing (fmem. bit) and sets the result into the carry flag.

XOR1 CY, fmem. bit

Function: $CY \leftarrow CY \nabla (\text{bit specified by operand})$

Exclusive-ORs the contents of the carry flag and the contents of the data memory bit specified by bit manipulation addressing (fmem. bit) and sets the result into the carry flag.

9.4.9 Branch Instructions

BR addr

Function: $PC_{10} \text{ to } PC_0 \leftarrow \text{addr}$; $\text{addr} = 000\text{H to } 77\text{FH}$

Branches to the address addressed by 11-bit immediate data addr .

This instruction is an assembler pseudo instruction. During assembly, the assembler automatically replaces this instruction with the optimum instruction from among the BRCB !caddr and BR \$addr instructions.

BR \$addr

Function: $PC \leftarrow \text{addr}$; $\text{addr} = (\text{PC}-15) \text{ to } (\text{PC}-1), (\text{PC}+2) \text{ to } (\text{PC}+16)$

This is a relative branch instruction with a branch range of (-15 to -1) and (+2 to +16) from the current address. Page boundary and block boundary are not affected.

BRCB !caddr

Function: $PC_{10} \text{ to } PC_0 \leftarrow \text{caddr}$; $\text{caddr} = A_{10} \text{ to } A_0 : 000\text{H to } 77\text{FH}$

Branches to the address addressed by 11-bit immediate data caddr ($A_{10} \text{ to } A_0$).

9.4.10 Subroutine Stack Control Instructions

CALLF !faddr

Function: $(SP-1) \leftarrow PC_7 \text{ to } PC_4$, $(SP-2) \leftarrow PC_3 \text{ to } PC_0$,
 $(SP-3) \leftarrow 0, 0, 0, 0$
 $(SP-4) \leftarrow 0, PC_{10} \text{ to } PC_8$,
 $SP \leftarrow SP-4$, $PC \leftarrow A_{10} \text{ to } A_0$
 faddr = $A_{10} \text{ to } A_0$: 000H to 77FH

Saves the contents of the program counter (PC; return address) to the data memory (stack) addressed by the stack pointer (SP) and decrements the SP, then branches to the address addressed by 11-bit immediate data faddr.

The range that can be called is limited to addresses 000H to 77FH (0 to 1919).

RET

Function: $PC_{10} \text{ to } PC_8 \leftarrow (SP)$, $PC_3 \text{ to } PC_0 \leftarrow (SP+2)$,
 $PC_7 \text{ to } PC_4 \leftarrow (SP+3)$, $SP \leftarrow SP+4$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC), then increments the contents of the SP.

RETS

Function: $PC_{10} \text{ to } PC_8 \leftarrow (SP)$, $PC_3 \text{ to } PC_0 \leftarrow (SP+2)$
 $PC_7 \text{ to } PC_4 \leftarrow (SP+3)$, $SP \leftarrow SP+4$, Then skip unconditionally

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and increments the contents of the SP, then skips unconditionally.

RETI

Function: $PC_{10} \text{ to } PC_8 \leftarrow (SP)$, $PC_3 \text{ to } PC_0 \leftarrow (SP+2)$,
 $PC_7 \text{ to } PC_4 \leftarrow (SP+3)$, $PSW_L \leftarrow (SP+4)$, $PSW_H \leftarrow (SP+5)$, $SP \leftarrow SP+6$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the program counter (PC) and program status word (PSW), then increments the SP.

This instruction is used to return from an interrupt handling routine.

PUSH rp

Function: $(SP-1) \leftarrow rp_H, (SP-2) \leftarrow rp_L, SP \leftarrow SP-2$

Saves the contents of register pair rp (XA, HL) to the data memory (stack) addressed by the stack pointer (SP), then decrements the SP.

The high-order side (rp_H : X, H) of the register pair is saved to the stack addressed by (SP-1) and the low-order side (rp_L : A, L) is saved to the stack addressed by (SP-2).

POP rp

Function: $rp_L \leftarrow (SP), rp_H \leftarrow (SP+1), SP \leftarrow SP+2$

Restores the contents of the data memory (stack) addressed by the stack pointer (SP) to register pair rp (XA, HL), then increments the SP.

The contents of (SP) are restored to the low-order side (rp_L : A, L) of the register pair and the contents of (SP+1) are restored to the high-order side (rp_H : X, H).

9.4.11 Interrupt Control Instructions

EI

Function: $\text{IME} \leftarrow 1$

Sets the interrupt master enable flag (1), and enables interrupts. Whether or not interrupts are accepted is determined by each interrupt enable flag.

EI IEXXX

Function: $\text{IE}_{\text{xxx}} \leftarrow 1$; $\text{xxx} = \text{N}_2$ to N_0

Sets the interrupt enable flag (IE_{xxx}) (1), and enables the interrupt. ($\text{xxx} = \text{BT}, \text{CSI}, 0, 2$)

DI

Function: $\text{IME} \leftarrow 0$

Resets the interrupt master enable flag and disables interrupts regardless of the contents of each interrupt enable flag.

DI IEXXX

Function: $\text{IE}_{\text{xxx}} \leftarrow 0$; $\text{xxx} = \text{N}_2$ to N_0

Resets the interrupt enable flag (IE_{xxx}) (0), and disables the interrupt. ($\text{xxx} = \text{BT}, \text{CSI}, 0, 2$)

9.4.12 Input/Output Instructions

IN A, PORTn

Function: $A \leftarrow \text{PORTn}$; $n = N_3 \text{ to } N_0 : 0 \text{ to } 3, 5, 6$

Transfers the contents of the port specified by PORTn ($n = 0 \text{ to } 3, 5, 6$) to the A register.

Note Only 0 to 3, 5 or 6 can be specified at n.

Output latch data (output mode) or pin data (input mode) is fetched according to input/output mode specification.

OUT PORTn, A

Function: $\text{PORTn} \leftarrow A$; $n = N_3 \text{ to } N_0 : 2, 3, 5, 6$

Transfers the contents of the A register to the output latch of the port specified by PORTn ($n = 2, 3, 5, 6$)

Note Only 2, 3, 5, or 6 can be specified at n.

9.4.13 CPU Control Instructions

HALT

Function: $PCC. 2 \leftarrow 1$

Sets the HALT mode (This instruction sets bit 2 of the processor clock control register.).

Note The instruction following the HALT instruction is made an NOP instruction.

STOP

Function: $PCC. 3 \leftarrow 1$

Sets the STOP mode (This instruction sets bit 3 of the processor clock control register.).

Note The instruction following the STOP instruction is made an NOP instruction.

NOP

Function: Expend one machine cycle without performing any operation.

APPENDIX A. TABLE OF INSTRUCTION USABLE WITH EVAKIT-75X ONLY

Since EVAKIT-75X (75X series common evaluation board) supports the 75X series functions, it can execute the following instructions not available with the μ PD75402A. Since the μ PD75402A and μ PD75P402 cannot execute these instructions even though they can be executed on the EVAKIT, do not use them.

Mnemonic	Operands	Mnemonic	Operands	
MOV	A, reg	ADDC	rp'1, XA	
	A, @rpa		XA, rp'	
	reg1, A		A, reg	
	reg1, #n4		reg, A	
	rp'1, XA		XA, @HL	
	XA, rp'		@HL, XA	
	XA, @HL	SUBS, SUBC	A, @HL	
	@HL, XA		rp'1, XA	
XCH	A, @rpa		XA, rp'	
	HL, mem		A, reg	
	XA, @HL		reg, A	
	XA, rp'		XA, @HL	
MOVT	XA, @PCDE		@HL, XA	
	XA, @BCDE		AND, OR, XOR	A, #n4
	XA, @BCXA			rp'1, XA
MOV1	CY, fmem. bit		XA, rp'	
	CY, pmem. @L		mem, A	
	CY, @H + mem. bit		A, reg	
	fmem. bit, CY		reg, A	
	pmem. @L, CY		XA, @HL	
	@H + mem. bit, CY		@HL, XA	
ADDS	rp'1, XA	ROLC	rp	
	XA, rp'		A	
	XA, #n8	RORC	rp	
	A, reg	INCS	rp1	
	reg, A		@HL	
	XA, @HL		XA	
	@HL, XA		DECS	rp'

APPENDIX A. TABLE OF INSTRUCTION USABLE WITH EVAKIT-75X ONLY

Mnemonic	Operands	Mnemonic	Operands
DECS	mem	NOT1	fmem. bit
	@HL		pmem. @L
SKE	A, reg		BR
	XA, rp'	!addr	
	XA, @HL	PCDE	
	@HL, #n4	PCXA	
	A, mem	BCDE	
SET1, CLR1, SKF, SKT, SKTCLR	pmem. @L	CALL	BCXA
	@H + mem. bit		!addr
AND1, OR1	CY, pmem. @L	PUSH	BS
	CY, @H + mem. bit	POP	BS
	CY,/fmem. bit	IN	XA, PORTn
	CY,/pmem. @L	OUT	PORTn, XA
	CY,/@H + mem. bit	SEL	MBn
XOR1	RBn		
	CY, @H + mem. bit	GETI	taddr

reg : X, A, B, C, D, E, H, L
 rp : XA, BC, DE, HL
 rp' : XA, BC, DE, HL, XA', BC', DE', HL'
 rpa : HL+, HL-, DE, DL

APPENDIX B. DEVELOPMENT TOOLS

★

The following development tools are available for system development using the μ PD75402A:

Language Processor

RA75X relocatable assembler	Host Machine		OS		Supply Medium		Ordering Code (Product Name)	
	PC-9800 series		MS-DOS™ (Ver. 3.30 Ver. 5.00 A*)		3.5-inch 2HD		μ S5A13RA75X	
					5-inch 2HD		μ S5A10RA75X	
	IBM PC/AT™		PC DOS™ (Ver. 3.1)		5-inch 2HC		μ S7B10RA75X	

PROM Writing Tools

Hardware	PG-1500		This PROM programmer allows programming, in standalone a mode or via operation from a host computer, of a singlechip microcomputer with on-chip PROM by connection of the d board provided and a separately available programmer adapter. It can program typical 256K-bit to 4M-bit PROMs.							
	PA-75P402CT PA-75P402GB		PROM programmer adapter for μ PD75P402C/CT/GB, used connected to the PG-1500. <ul style="list-style-type: none"> • PA-75P402CT ... For μPD75P402C/CT • PA-75P402GB ... For μPD75P402GB 							
Software	PG-1500 controller		Connects PG-1500 and host machine via a serial and parallel interface, and controls the PG-1500 on the host f machine.							
			Host Machine		OS		Supply Medium		Ordering Code (Product Name)	
			PC-9800 series		MS-DOS (Ver. 3.30 Ver. 5.00 A*)		3.5-inch 2HD		μ S5A13PG1500	
							5-inch 2HD		μ S5A10PG1500	
IBM PC/AT		PC DOS (Ver. 3.1)		5-inch 2HC		μ S7B10PG1500				

* The task swap function, which is provided with Ver.5.00/5.00A, is not available with this software.

Remarks Operation of the relocatable assembler and PG-1500 controller is guaranteed only on the host machines and operating systems quoted above.

Debugging Tools

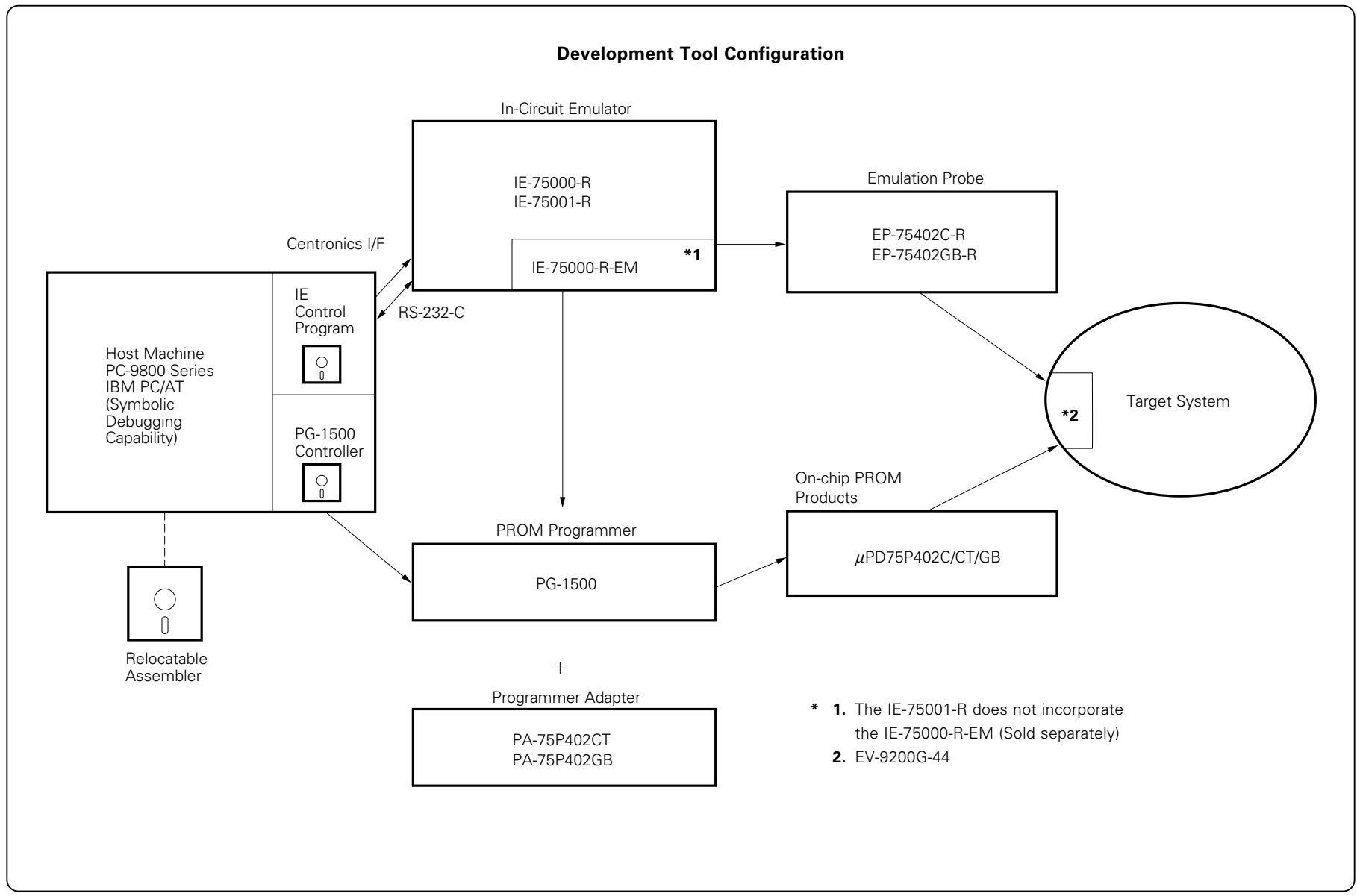
The following in-circuit emulators (IE-75000-R and IE-75001-R) are available as the μ PD75402A program debugging tools.

Their respective system configurations are as follows.

Hardware	IE-75000-R*1	The IE-75000-R is an in-circuit emulator for hardware/software debugging in development of an application system using the 75X series. Used in combination with an emulation probe. Connected with a host machine and PROM programmer, the IE- 75000-R can perform efficient debugging.			
	IE-75000-R-EM	Emulation board for evaluation of an application system using the 75X series. Used in combination with the IE-75000-R or IE-75001-R. Incorporated in the IE-75000-R.			
	IE-75001-R*2	The IE-75001-R is an in-circuit emulator for hardware/software debugging in development of an application system using the 75X series. Used in combination with the separately available IE-75000-R- EM emulation board and emulation probe. Connected with a host machine and PROM programmer, the IE- 75001-R can perform efficient debugging.			
	EP-75402C-R	μ PD75402AC/75402ACT emulation probe. Used connected with the IE-75000-R, IE-75001-R or IE-75000-R- EM.			
	EP-75402GB-R	μ PD75402AGB emulation probe. Used connected with the IE-75000-R, IE-75001-R or IE-75000-R- EM Provided with the EV-9200G-44, 44-pin conversion socket which facilitates connection with the target system.			
Software	IE control program	Connects the IE-75000-R or IE-75001-R with a host machine via RS-232-C or Centronics I/F and controls the IE-75000-R or IE-75001-R on the host machine.			
		Host Machine	OS	Supply Medium	Ordering Code (Product Name)
		PC-9800 series	MS-DOS (Ver. 3.30 λ Ver. 5.50 A*3)	3.5-inch 2HD	μ S5A13IE75X
				5-inch 2HD	μ S5A10IE75X
	IBM PC/AT	PC DOS (Ver. 3.1)	5-inch 2HC	μ S7B10IE75X	

- * 1. Maintenance product
- 2. IE-75000-R-EM is sold separately.
- 3. The task swap function, which is provided with Ver.5.00/5.00A, is not available with this software.

Remarks Operation of the IE control program is guaranteed only on the above quoted host machines and operating systems.



APPENDIX C. MASK ROM ORDERING PROCEDURE

When completing the μ PD75402A program and ordering the mask ROM, proceed as follows:

① Mask ROM order reservation

Provide us with the mask ROM ordering schedule through your dealer or our sales department (If we are not informed in advance, processing may be delayed.).

② Preparation of ordering medium

The medium for mask ROM order is UV-EPROM or 8-inch IBM format floppy disk. When ordered with UV-EPROM, please prepare three UV-EPROMs with the same contents. (Send the mask options data filled in the mask option information documents.)

③ Preparation of the necessary documents

When ordering the mask ROM, please fill in the following documents:

- Mask type ROM order form
- Mask type ROM order check sheet
- Mask option information documents

④ Send the medium prepared in ② and the documents described in ③ to us through your special agents or our sales department by the scheduled order date.

Note For details, refer to the information document "ROM Code Ordering Method" "Documents No. IEM-834".

APPENDIX D. INSTRUCTION INDEX (ALPHABETIC ORDER)

Instruction	Page	Instruction	Page
ADDC A, @HL	163	NOP	176
ADDS A, #n4	163	NOT A	165
ADDS A, @HL	163	NOT1 CY	168
AND A, @HL	163	OR A, @HL	164
AND1 CY, fmem. bit	170	OR1 CY, fmem. bit	170
BR addr	171	OUT PORTn, A	175
BR \$addr	171	POP rp	173
BRCB ! caddr	171	PUSH rp	173
CALLF ! faddr	172	RET	172
CLR1 CY	168	RETI	172
CLR1 fmem. bit	169	RETS	172
CLR1 mem. bit	169	RORC A	165
DECS reg	166	SET1 CY	168
DI	174	SET1 fmem. bit	169
DI IExxx	174	SET1 mem. bit	169
EI	174	SKE A, @H	167
EI IExxx	174	SKE reg, #n4	167
HALT	176	SKF fmem. bit	170
IN A, PORTn	175	SKF mem. bit	170
INCS mem	166	SKT CY	168
INCS reg	166	SKT fmem. bit	169
MOV A, mem	160	SKT mem. bit	169
MOV A, #n4	159	SKTCLR fmem. bit	170
MOV A, @HL	159	STOP	176
MOV mem, A	160	XCH A, @HL	161
MOV mem, XA	160	XCH A, mem	161
MOV rp, #n8	159	XCH A, reg1	161
MOV XA, mem	160	XCH XA, mem	161
MOV @HL, A	160	XOR A, @HL	164
MOVT XA, @PCXA	162	XOR1 CY, fmem. bit	170

APPENDIX E. HARDWARE INDEX (ALPHABETIC ORDER)

Hardware		Page	Hardware		Page
Symbol	Name		Symbol	Name	
ACKD	Acknowledge detect flag	80, 104, 109	IRQ2	INT2 interrupt request flag	129
ACKE	Acknowledge enable flag	80, 104, 108	IRQBT	BT interrupt request flag	129
ACKT	Acknowledge trigger bit	79, 104, 107	IRQCS	Serial interface interrupt request flag	129
BSYE	Sync busy enable bit	80, 105, 109	IST0	Interrupt status flag	133
BT	Basic interval timer	66	PC	Program counter	31
BTM	Basic interval timer mode register	67	PCC	Processor clock control mode register	55
CLOM	Clock output mode register	64	PMGA, PMGB	Port mode register (A, B)	47
CMDD	Command detect flag	79, 104, 106	POGA	Pull-up resistor specification register	51
CMDT	Command trigger bit	79, 87, 103, 106	PORT 0	Port 0 to 3, 5, 6	42
COI	Address comparator coincidence signal	76, 86, 102	PSW	Program status word	39
CSIE	Serial interface operation enable/disable specification bit	76, 83, 86, 102	RELD	Bus release detect flag	79, 104, 106
CSIM	Serial operating mode register	73, 74, 85, 101	RELT	Bus release trigger bit	79, 87, 103, 106
CY	Carry flag	39	SBIC	Serial bus interface control register	78, 87, 103, 106 to 109
IE0	INT0 interrupt enable flag	129	SIO	Serial I/O shift register	73, 81, 113
IE2	INT2 interrupt enable flag	129	SK0 to SK2	Skip flag	40
IEBT	BT interrupt enable flag	129	SP	Stack pointer	37
IECSI	Serial interface interrupt enable flag	129	SVA	Slave address register	73, 82, 113
IM0	INT0 mode register	131	WUP	Wake-up function specification bit	76, 85, 102
IME	Interrupt master enable flag	132			
IRQ0	INT0 interrupt request flag	129			